Verfahren und Vorrichtung zur Stimulation von Funktionen zur Steuerung von Betriebsabläufen

Stand der Technik

Die Erfindung geht aus von einem Verfahren und einer Vorrichtung zur Stimulation von Funktionen zur Steuerung von Betriebsabläufen, insbesondere bei einem Fahrzeug, gemäß den Oberbegriffen der unabhängigen Ansprüche. Ebenso geht die Erfindung von einem entsprechenden Steuergerät sowie einem entsprechenden Computer für die Funktionsstimulation und mit dem damit zusammenhängenden Computerprogramm als auch dem entsprechenden Computerprogrammprodukt mit den Merkmalen gemäß den Oberbegriffen der Ansprüche aus.

In der Funktionsentwicklung von Steuergerätesoftware, insbesondere bei Fahrzeug-Steuergeräten für die Motor-, Bremsen-, Getriebe- usw. -Steuerung, ist die Bypass-Anwendung ein Rapid-Prototypingverfahren um neue Steuergerätefunktionen zu entwickeln und zu testen. Eine solche Funktionsentwicklung ist aber auch bei allen anderen Steuergeräteanwendungen wie z.B. im Automatisierungs- und Werkzeugmaschinenbereich usw. möglich.

Als Entwicklungsverfahren kommen hierfür die beiden Anwendungen externer Steuergeräte-Bypass wie diese z.B.in der DE 101 06 504 A1 gezeigt ist sowie interner Steuergeräte-Bypass, wie z.B. in der DE 102 286 10 A1 offenbart zum Einsatz.

Dabei betrifft die DE 101 06 504 A1 ein Verfahren und eine Emulationsvorrichtung zum Emulieren von Steuer- und/oder Regelfunktionen eines Steuer- oder Regelgeräts insbesondere

eines Kraftfahrzeugs. Zum Emulieren werden die Funktionen in einen externen Emulationsrechner ausgelagert, wobei vor Beginn der Emulation über eine Software-Schnittstelle des Emulationsrechners und eine Software-Schnittstelle des Steuer-/ Regelgeräts eine Datenverbindung hergestellt wird. Um die Entwicklung und Programmierung neuer Steuer-/ Regelfunktionen des Steuer-/ Regelgeräts deutlich zu beschleunigen, wird vorgeschlagen, dass die Software-Schnittstellen zur Emulation von unterschiedlichen Steuer-/ Regelfunktionen vor Beginn der Emulation ohne Änderung der Software konfiguriert werden.

Die DE 102 286 10 A1 zeigt ihrerseits ein Verfahren und eine Vorrichtung zum Überprüfen eines Steuerprogramms mittels mindestens einer Bypassfunktion, bei dem das Steuerprogramm zusammen mit der mindestens einen Bypassfunktion auf einer elektrischen Recheneinheit ausgeführt wird. Die Ankopplung der Bypassfunktionen geschieht dabei durch dynamisches Linken an vorgegebenen Schnittstellen.

Unabhängig dieser beiden genannten Verfahren und Vorrichtungen werden für die Anwendbarkeit Eingriffe in der Steuergerätesoftware benötigt. Diese Eingriffe werden mit dem Begriff Bypass-Freischnitt oder Software-Freischnitt bezeichnet. Ein Bypass- bzw. Software-Freischnitt beschreibt genau die Stelle in einer Softwarefunktion, an der der Inhalt einer Steuergeräte-Variable nicht durch das Softwareprogramm, sondern über Umwege z.B. über eine Bypass-Softwarefunktion beschrieben wird. Software-Freischnitte sind sehr individuell und im Normalfall nicht Bestandteil eines Steuergeräte-Softwareprogrammes, da hierfür Speicher-Ressourcen verbraucht werden.

Benötigt ein Funktionsentwickler ein Steuergeräte-Programm mit Software-Freischnitten, so werden diese erst nach Beauftragung der Entwicklungsabteilung in einen Programmstand eingebaut. Die Softwareentwicklung ändert hierzu manuell den Sourcecode der entsprechenden Funktion und erstellt über einen Compiler- und Linklauf ein neues Steuergeräte-Programm, welches explizit für die Prototypinganwendung zum Einsatz kommt.

Der Nachteil des Verfahrens bzw. der Vorrichtung wie im Stand der Technik ausgeführt liegt in der langen Durchlaufszeit bis hin zur Verfügbarkeit des Rapid-Prototyping-Programmstandes. Ein wesentlicher Faktor ist hierbei der damit verbundene hohe technische und administrative Aufwand zur Spezifikation und Umsetzung der Software-Eingriffe.

Nach aktuellem Wissensstand basiert ein vergleichbares Verfahren auf der Idee, nur die Store-Befehle (Schreibzugriff auf eine Steuergeräte-Variable), durch Sprungbefehle auf eine Unterfunktion zu ersetzen. Bei Mikrocontrollern mit gemischtem Befehlssatz (16-/32-Bit-CPU-Befehle) können aber die Store-Befehle 16-Bit-breit sein, da die Adressierung indirekt über Adressregister erfolgt. Diese 16-Bit-breiten Befehle können für den Aufruf einer Unterfunktion nicht herangezogen werden, da der direkte, adressorientierte Aufruf einer Unterfunktion einen 32-Bit-breiten Sprungbefehl erfordert. Somit ist das Verfahren im Stand der Technik dann nur bedingt einsetzbar und kann nur bei Mikroprozessoren mit reinem 32-Bit-Befehlssatz angewendet werden. D.h. bei festgelegter Bitbreite des Storebefehls ist die Flexibilität bezüglich der Funktionsentwicklung hier stark eingeschränkt. Dies gilt auch wenn ein bestimmter Storebefehl aus anderen Gründen überhaupt nicht manipuliert werden darf, so dass eine solche Belegung durch Sprungbefehl auf eine Unterfunktion dann nicht möglich ist.

In der bereits erwähnten Funktionsentwicklung für Steuergerätesoftware wird das Testen von Softwarefunktionen an Labor-Autos (Messplätze) oder über entsprechende Softwaretestumgebungen durchgeführt (Modultest). Beim Testen an Labor-Autos werden die Steuergeräteeingänge von aussen, durch elektrische Signale stimuliert um so die entsprechenden Reaktionen des Steuergerätes und der Software zu beobachten und zu messen. Bei den verfügbaren Softwaretestumgebungen für den Modultest, werden ausschliesslich die zu untersuchenden Softwarefunktionen in einem Standalone-Betrieb innerhalb der Testumgebung betrachtet ohne die Rückwirkung oder Querkopplung zum Gesamtsystem zu berücksichtigen.

Es ist Aufgabe der Erfindung Softwarefunktionen im einem integrierten Zustand innerhalb eines Softwareprogrammes zur Laufzeit zu stimulieren und zu testen. und damit die genannten Probleme im Stand der Technik zu überwinden.

Vorteile der Erfindung

Die Erfindung geht aus von einer Vorrichtung und einem Verfahren zur Stimulation von Funktionen zur Steuerung von Betriebsabläufen, wobei die Funktionen auf wenigstens eine globale Variable wenigstens eines Programms zur Steuerung zurückgreift, wobei vorteilhafter Weise wenigstens eine Stimulationsfunktion vorgesehen ist, die auf die wenigstens eine

globale Variable über wenigstens einen Softwarefreischnitt zugreift.

Dabei erfolgt zweckmäßiger Weise die Stimulation der Funktionen innerhalb des Programms zur Laufzeit des Programms, insbesondere in Echtzeit innerhalb eines Laufzeitsystems.

Von Vorteil ist dabei, dass der globalen Variable eine erste Dateninformation zugeordnet ist und diese erste Dateninformation durch eine zweite Dateninformation, die neuen Stimulationswerten entspricht ersetzt wird.

Bevorzugt im Rahmen der Erfindung ist es, dass die Stimulation der Funktionen durch einen internen Bypass erfolgt, wobei der Softwarefreischnitt dadurch erfolgt, dass der globalen Variable eine Adressinformation zugeordnet ist, wobei die Adressinformation durch einen Ladebefehl aus einem Speichermittel geladen wird und die Adressinformation der globalen Variable des Ladebefehls ersetzt wird.

Vorteilhafter Weise wird dabei die Adressinformation der globalen Variable durch die Adressinformation einer Zeigervariable ersetzt, wobei aus der Adressinformation eine Anfangsadresse der Funktion ermittelt wird und die Funktionen zur Steuerung von Betriebsabläufen durch Ersetzen der Adressinformation durch Zusatzfunktionen ersetzt werden.

Zweckmäßig kann der Softwarefreischnitt auch dadurch erfolgen, dass die globale Variable durch einen Speicherbefehl angesprochen wird und der Speicherbefehl auf die globale Variable manipuliert wird, indem der Speicherbefehl durch einen Sprungbefehl ersetzt wird und die Funktionen zur Steuerung der Betriebsabläufe durch Ersetzen des Speicherbefehls durch den Sprungbefehl durch Zusatzfunktionen ersetzt werden.

Bei der Erfindung wird somit zweckmäßiger Weise ein „dynamisches Anhängen" („Dynamic-Hooks") von Software-Freischnitten ohne Source-Codeänderungen verwendet. Die dazu beschriebenen Methoden ändern die Adressinformation von Ladebefehlen oder „Load"-Befehlen, ändert Funktionsaufrufe und fügt neue Programmcodes hinzu. Diese Änderungen werden an einem vorhandenen Softwareprogrammstand z.B. auf der Basis von gezielten HEX-Code-Modifikationen durchgeführt.

Von Vorteil ist weiterhin, dass die Adressinformation der globalen Variable durch die Adressinformation einer Zeigervariable ersetzt wird, wobei die Adressinformation der Zeigervariable in einem reservierten Speicherbereich, insbesondere des Speichermittels im Steuergerät, vorliegt.

Zusätzlich zur Modifikation bezüglich der Ladebefehle ist in einer Ausgestaltung zweckmäßiger Weise vorgesehen dass ein Speicherbefehl oder Store-Befehl auf die globale Variable manipuliert wird, indem der Speicherbefehl durch einen Sprungbefehl ersetzt wird. Dabei werden zweckmäßiger Weise die Funktionen zur Steuerung der Betriebsabläufe durch Ersetzen des Speicherbefehls durch den Sprungbefehl durch Zusatzfunktionen ersetzt und/oder erweitert.

Entsprechend der genannten Vorrichtung sowie des Verfahrens ist durch die Erfindung ein eine solche Vorrichtung enthaltendes Steuergerät ebenso offenbart und Gegenstand der Erfindung wie ein zur Ausführung eines solchen Verfahrens geeignetes Computerprogramm. Dieses Computerprogramm wird dazu auf einem Computer, insbesondere einer erfindunggemäßen Applikationssteuergeräteanordnung oder auch einem Applikations-PC ausgeführt. Dabei ist das erfindungsgemäße Computerprogramm auf einem beliebigen maschinenlesbaren Träger abspeicherbar. Ein solcher computerlesbarer Datenträger oder maschinenlesbarer Träger kann insbesondere eine Diskette, eine CD-ROM, eine DVD, ein Memory-Stick oder auch jedes andere mobile Speichermedium sein. Ebenso sind Speichermedien wie ROM, PROM, EPROM, EEPROM oder Flash-Speicher sowie flüchtige RAM Speicher usw. zur Abspeicherung möglich. Die Wahl des Speichermediums bzw. des maschinenlesbaren Trägers ist somit nicht einschränkend im Hinblick auf das Computerprogramm-Produkt als Gegenstand der Erfindung zu sehen.

Mit der Erfindung können die verschiedenen Rapid Prototyping-Verfahren, Software-Testverfahren und Datenkalibrier-Verfahren schneller einsetzbar und flexibler handhabbar gemacht werden. Somit erfolgt die Durchführung der Software-Freischnitte ohne Bindung von Software-Entwicklungskapazität. Damit ergibt sich insgesamt ein geringerer technischer und administrativer Aufwand, somit Reduzierung der Kosten. Gleichzeitig können Mikroprozessortypen mit einem gemischten Befehlssatz von z.B. 16-/32-Bit- breiten CPU-Befehlen unterstützt werden.

Beim Aktivieren der Testabläufe wird die zu untersuchende Steuergerätefunktion vorteilhafter Weise durch die integrierte Testsoftware innerhalb des Laufzeitsystems selbständig stimuliert.

Die Vorteile des erfindungsgemäßen Gegenstandes liegen auch darin, dass zur Stimulation der Steuergerätefunktion keine elektrischen Signale an das Steuergerät von aussen angelegt werden müssen. Ein weiterer Vorteil ist der, dass die berechneten Ausgangswerte der stimulierten Steuergerätefunktion, dem Echtzeitsystem direkt wieder zur Verfügung stehen. Somit können z.B. komplexe Regelalgorithmen innerhalb des Steuergerätes genauer untersucht, getestet und ggf. weiterentwickelt werden.

Weitere Vorteile und Vorteilhafte Ausgestaltungen ergeben sich aus der Beschreibung sowie den Merkmalen der Ansprüche.

Zeichnung

Die Erfindung wird im Weiteren anhand der in den Figuren dargestellten Gegenständen näher erläutert. Dabei zeigt Figur 1 eine erfindungsgemäße Anordnung oder Vorrichtung zur Anpassung der Funktionen.

Figur 2 offenbart den Ablauf zur Ermittlung der Freischnittstellen oder Softwarefreischnitte im Programm.

Figur 3 zeigt eine Übersicht und Auswahl verschiedener Verfahren zur Modifikation der Load- und/oder Store-Befehle.

Figur 4 zeigt für ein erstes und bevorzugtes Modifikationsverfahren des Ladebefehls oder Load-Befehls eine Programmdarstellung.

Figur 5 offenbart für ein zweites Modifikationsverfahren des Speicherbefehls oder Store-Befehls eine Programmdarstellung.

Figur 6 offenbart für ein drittes Modifikationsverfahren des Speicherbefehls oder Store-Befehls eine Programmdarstellung.

Figur 7 offenbart für ein viertes Modifikationsverfahren des Speicherbefehls oder Store-Befehls eine Programmdarstellung.

Figur 8 zeigt eine Prinzipdarstellung zur Anpassung der Aufrufe der Funktionen zur Steuerung der Betriebsabläufe.

Die Hook-Funktion zur Einbindung der Zusatzfunktionen wird in Figur 9 gezeigt.

Figur 10 zeigt eine schematische Darstellung der Speichersegmente im Speichermittel bezüglich der Hook-Funktion.

Ein vollständiger Entwicklungsprozess gemäß dieser Erfindung ist schließlich in Figur 11 näher dargestellt.

Figur 12 zeigt den internen Bypass, insbesondere den Vorgang des dynamischen Linkens.

Figur 13 offenbart eine Prinzipdarstellung zur Stimulation der Funktionen.

Figur 14 zeigt einen Ablaufplan zur Erstellung der Testablaufsoftware.

In Figur 15 schließlich ist der erfindungsgemäße Entwicklungsprozess gezeigt.

Beschreibung der Ausführungsbeispiele

Figur 1 zeigt in schematischer Darstellung eine Applikationsanordnung mit einem Steuergerät 100 und einem Applikationssystem 101, welche über eine Verbindung 102 mit den Schnittstellen 103 und 104 gekoppelt sind. Diese Verbindung 102 kann dabei ebenso leitungsgebunden wie leitungslos ausgeführt sein. Mit 105 ist ein Mikroprozessor, insbesonder mit gemischtem Befehlssatz dargestellt. 106 zeigt ein Speichermittel, welches ein Adressregister 108, ein Datenregister 107 sowie einen Speicherbereich für das wenigstens eine bezüglich der Funktionen anzupassende Programm enthält. Das Kontrollmittel zur Realisierung der Erfindung kann dabei im Aplikationssystem enthalten sein bzw. durch dieses repräsentiert werden oder aber unter Benutzung des Mikroprozessors selbst ausgebildet sein. Ebenso können Speichermittel zur Realiserung der Erfindung außerhalb des Steuergerätes eben insbesondere im Applikationssystem untergebracht sein. Mit der dargestellten Vorrichtung ist der erfindungsgemäße Gegenstand realisierbar.

Zwar kann die Anpassung der Funktionen mit externem Bypass erfolgen, die vorteilhafte Ausgestaltung aber ist die Anpassungen intern derart vorzunehmen, dass diese im Programmlauf eingebunden sind und so ein dynamisches Anhängen (Dynamic-Hooks) von Softwareeingriffen ohne Sourcecode-Änderungen erfolgt.

Der hier beschriebene Gegenstand ändert die Adressinformation von Load-Befehlen, ändert den Inhalt von Store-Befehlen, ändert die Adressinformation von Funktionsaufrufen und fügt neue Programmcodes hinzu. Diese Änderungen werden hier im Ausführungsbeispiel an einem vorhandenen Softwareprogrammstand auf der Basis von gezielten Hexcode-Modifikationen durchgeführt.

Die verschiedenen Bestandteile bezüglich des erfindungsgemäßen Gegenstandes "Dynamischer Software-Freischnitt" die im weiteren offenbart werden lauten wie folgt:

- Die Ermittlung der Programmstellen
- Die Modifikation der Programmstellen, mit
  - der Modifikation der Load-/Store-Befehle und
  - der Modifikation der Funktionsaufrufe
- Die Erstellung von zusätzlichem Programmcode
- Das Einbinden des Software-Freischnittcodes
- Die Segmentierung der Speicherbereiche und
- Der Entwicklungsprozess zur Erstellung des Programmcodes
- Interner Bypass
- Softwaretest durch Stimulation der Funktionen

Das nachfolgend dargestellte Verfahren basiert auf dem Einsatz von Mikrocontrollern, deren Befehlsatz gemischt ist und insbesondere 16-/32-Bit-breite CPU-Befehle umfasst. Als exemplarisches Beispiel dient hier z.B. der Mikrocontroller TriCore TC17xx (RISC/DSP/CPU) von Infineon, welcher Bestandteil eines Steuergeräts zur Steuerung von Betriebsabläufen, insbesondere bei einem Fahrzeug, z.B. zur Motorsteuerung oder zur Steuerung von Lenkung, Getriebe, Bremse usw. ist.

Das Verfahren kann aber auch bei Mikroprozessoren mit nicht gemischtem Befehlssatz angewendet werden, insbesondere bei reinen 32-Bit-Mikroprozessoren (RISC- Prozessoren, z.B. PowerPC/ MPC5xx).

Grundsätzlich wird bei dem Verfahren davon ausgegangen, dass der Codegenerator des Compilers die Maschinen-Befehle sequentiell anordnet. Darunter versteht man die

aufeinanderfolgende Befehlsanordung zum Laden von Adressinformationen z.B. einer indirekt adressierten Steuergeräte-Variable, in entsprechende Adressregister. Im Gegensatz hierzu befindet sich bei einer direkt adressierten Variable die Adressinformation im Befehl selbst. Dieser Sachverhalt ist bei den meisten Compilern gegeben.

Die Ermittlung der Programmstellen (Figur 2)

Ausgangspunkt hierfür ist ein Steuergeräte-Softwareprogramm, welches z.B. in Form einer Hexcode-Datei zur Verfügung steht. Als weitere Dateien dienen eine Datenbescheibungsdatei (z.B. ASAP)
und eine Linker-Datei (z.B. ELF-Binär), welche Informationen über Steuergeräte-Variablen und Steuergeräte-Funktionen liefern.

Mit einem Disassembler-Softwareprogramm (z.B. ein Windows-Softwareprogramm) wird die Hexcode-Datei disassembliert. Die entsprechenden Adressen der freizuschneidenden Steuergeräte-Variablen werden aus der Datenbeschreibungsdatei oder aus einer für das Verfahren erstellten Referenzdatenbank entnommen.

Das erfindungsgemäß erstellte Disassembler-Programm z.B. ein Windows-Softwareprogramm sucht im disassemblierten Programmcode, unter Zuhilfenahme der Adressinformation der gesuchten Steuergeräte-Variable, die entsprechenden Zugriffbefehle auf diese Variable (Load-/Store-Befehle), welche sich auf den Variableninhalt auswirken.

Dieses Disassembler-Programm als Windows-Softwareprogramm ist ein Simulationsprogramm, welches die Registerinhalte nach jedem Assemblerbefehl überprüft. Wird ein Store-Befehl lokalisiert und entspricht der Inhalt des geladenen Adressregisters dem Adresswert der gesuchten Steuergeräte-Variable oder entspricht das Speicherziel des Store-Befehls der Variablenadresse, dann liegt eine Fundstelle vor, an der der Inhalt der Steuergeräte-Variable verändert wird.

Die Art und Weise, wie der Programmcode an den Fundstellen geändert wird, hängt von der jeweiligen Adressierungsart der Steuergeräte-Variable ab.

Dies ist in Figur 2 dargestellt. Darin ist mit 201 der Steuergeräte-Programmcode dargestellt. 202 zeigt die Softwarefunktion. Die Pfeile 203 symbolisieren das beschriebene Verfahren zur Ermittlung der Speicher- oder Store-Befehle. Mit 204 ist der Store-Befehl eines Variablenzugriffs dargestellt und zwar derart, dass bei direkter Adressierung das Speicherziel des Store-Befehls die RAM-Adresse ist und bei indirekter Adressierung der Inhalt des Adressregisters der RAM-Adresse entspricht, wodurch dann die Load- oder Lade-Befehle ermittelt werden können. Die Pfeile, die mit 205 bezeichnet sind symbolisieren dann das beschriebene Verfahren zur Ermittlung der Load-Befehle. Die Ladebefehle oder Load-Befehle zum Laden der Variablen-Adresse, speziell der globalen Variable sind mit 206 bezeichnet.

Die Modifikation der Programmstellen (Figuren 3 bis 8)

Dabei werden zum einen entsprechend unterschiedlicher Adressierungsarten die Load- Befehl Fundstellen und/oder die Store-Befehl-Fundstellen lokalisiert und für diese Fundstellen wird dann die Steuergeräte Funktion ermittelt in der sich die Fundstellen befinden, damit im gesamten Programmcode alle Funktionsaufrufe durch Funktionsaufrufe der neu erstellten Hook-Funktion(en) erfolgen können, so dass der ursprüngliche Funktionsaufruf der Steuergeräte-Funktion innerhalb der entsprechenden Hook-Funktion erfolgen kann.

Die Modifikation der Load/Store-Befehle

Bei dem beschriebenen Mikrocontroller(n) gibt es eine Vielzahl unterschiedlicher Adressierungsarten in den verschiedensten Ausführungen. Diese Vielfalt kann auf ein Minimum reduziert werden.

Nachfolgend sind vier Methoden dargestellt, die weitgehend mögliche Kombinationen eines schreibenden Zugriffs auf globale Variablen abdecken. Weitere Methoden zur Code-Analyse sind denkbar, wie z.B. eine relative Adressierung über vorbelegte Adressregister.

Dazu ist in Figur 3 eine Übersicht über die unterschiedlichen Methoden zur Modifikation der Load- und/oder Store- Befehle gegeben. Die Store-Befehle st.x bedeuten darin: st.b = store byte, st.h = store halfword und st.w = store word. Die in Figur 3 aufgeführten vier Methoden werden nachfolgend näher erläutert.

Die Modifikation der Programmstellen nach Methode 1 ist in Figur 4 näher dargestellt. Bei der Methode 1 handelt es sich beispielhaft um einen 16-Bit Store-Befehl und eine indirekten Adressierung. Ausgehend von der Position des gefundenen Store-Befehls, werden im disassemblierten Programmcode die Stellen zurückverfolgt, bis die zugehörenden Load-Befehle ermittelt sind. Für diese Methode sind die gefundenen Load-Befehle entscheidend. Diese Methode findet nicht nur Anwendung bei Problemen bei gemischtem Befehlssatz sondern auch wenn aus anderen Gründen der Ersatz des Speicher- oder Store-Befehls durch einen Sprungbefehl nicht möglich ist.

Bei der Methode 1 werden die ermittelten Load-Befehle, bezogen auf den nachfolgenden Store-Befehl, durch Adressinformationen einer Zeigervariable ersetzt. Diese Zeigervariable wird über eine Entwicklungsumgebung, insbesondere die DHooks-Entwicklungsumgebung erzeugt. Die Adresse der Zeigervariable befindet sich in einem reservierten Freibereich des Speichermittels, des Speicher-Layouts für Variablen. Die modifizierten Load-Befehle adressieren das gleiche Adressregister wie die Originalbefehle. Der Unterschied der geänderten Load-Befehle liegt in der Adressierungsart des Adressregisters und in der Adressinformation.

So ist in Figur 4 in einer Prinzipdarstellung die Methode 1 erläutert. Dabei ist mit 401 der ursprüngliche Programmcode und mit 411 der modifizierte Programmcode bezeichnet. 402 und 406 sowie 417 und 407 bilden die Steuergeräte-Funktion, hier function_a(). Dabei sind in 402 bzw. in 417 die Befehle bzw. Befehlssequenzen dargestellt und in 406 bzw. 407 die eigentliche Funktionalität. Mit axx% ist ein Zugriff auf ein Adressregister (z.B. a0 bis a15 bei 16 Bit Breite) dargestellt und mit dxx% ein Zugriff auf ein Datenregister (z.B. d0 bis d15 bei 16 Bit Breite). Dazu sind nun die Befehle movh.a und ld.a (Load-Befehle) sowie st.x (Store-Befehl) in 408, 409, 410, 413, 414 und 415 betrachtet. movh.a und ld.a sind in diesem Beispiel als 32Bit Befehle (siehe 412 und 403) dargestellt. Der Speicherbefehl st.x ist als 16 Bit Befehl (siehe 405) dargestellt und damit in diesem Beispiel durch einen 32Bit Sprungbefehl nicht ersetzbar. Wie gesagt gilt dies auch für alle anderen Fälle in denen ein solcher Ersatz nicht möglich oder nicht gewünscht ist. Der neue erfindungsgemäße Befehlscode bzw. Programmcode 403 wird nun auf 412 eingespielt und die Load Befehle werden auf die Zeigervariable iB_PtrMsg_xxx (Ptr = Pointer) geändert. Die Adresse der Steuergeräte Variable wird durch die Adresse der Zeigervariable gemäß 404 ersetzt. Das Verfahren zur Erstellung von zusätzlichem Programmcode bzw. Zusatzfunktionen wird später nach den vier Methoden ausführlich erläutert.

Die Modifikation der Programmstellen nach Methode 2 ist in Figur 5 näher dargestellt. Dabei gelten wie auch für alle übrigen Methodenbeispiele die gleichen Bezeichnungen und Abkürzungen wie für Methode 1. Dabei ist mit 501 der ursprüngliche Programmcode und mit 511 der modifizierte Programmcode bezeichnet. 502 und 506 sowie 517 und 507 bilden die Steuergeräte-Funktion, hier function_a(). Dabei sind in 502 bzw. in 517 die Befehle bzw. Befehlssequenzen dargestellt und in 506 bzw. 507 die eigentliche Funktionalität. Mit %axx ist ein Zugriff auf ein Adressregister (z.B. a0 bis a15 bei 16 Bit Breite) dargestellt und mit %dxx ein Zugriff auf ein Datenregister (z.B. d0 bis d15 bei 16 Bit Breite). Dazu sind nun wieder die Befehle movh.a und ld.a (Load-Befehle) sowie st.x (Store-Befehl) betrachtet. Der Speicherbefehl st.x ist nun als 32 Bit Befehl (siehe 505) dargestellt und damit in diesem Beispiel durch einen 32Bit Sprungbefehl jla ersetzbar. Der neue erfindungsgemäße Befehlscode bzw. Programmcode 503 (jla: Sprungbefehl) wird nun auf 505 eingespielt.

Bei der Methode 2 handelt es sich um einen 32-Bit Store-Befehl in Verbindung mit einer indirekten Adressierung. Der 32-Bit Store-Befehl wird durch einen absoluten Sprungbefehl jla 512 auf eine Software-Balkonfunktion (balcony_M2) ersetzt (siehe 520 Aufruf der Software Balkonfunktion). Bei dem jla-Sprungbefehl wird die Rücksprungadresse in Adressregister a11 abgelegt (siehe in 521).

In der genannten Software-Balkonfunktion 521 wird der Inhalt des Adressregister %axx, über das die Steuergeräte-Variable adressiert wird durch den Adresswert einer Zeigervariable (iB_PtrMsg_xxx) ersetzt. Der Index des Adressregisters %axx und des zuvor geladenen Datenregister %dxx ist in der Software-Balkonfunktion 521 identisch.

Werden 32-Bit-breite Store-Befehle für den Software-Freischnitt herangezogen, so wird hierzu zusätzlicher Programmcode benötigt. Dieser Programmcode wird in der DHooks-Entwicklungsumgebung erzeugt und als Balkon-Funktion bezeichnet. Die Balkon-Funktionen beinhalten zusätzliche Initialisierungs-, Kopier- und Freischnitt-Mechanismen und dienen als Software-Funktionen zur Erweiterung der Freischnittfunktionalität. Balkon-Funktionen werden für die Freischnitt-Methoden 2, 3 und 4 verwendet.

Durch den Sprungbefehl jla bleibt der Inhalt des verwendeten Datenregisters %dxx unverändert. In der Software-Balkonfunktion erfolgt nun die Adressierung über den Zeiger und somit die

Umlenkung des Store-Befehls auf die Zeigervariable. Der Store-Befehl st.x schreibt die Daten wie im Originalcode.

Anschliessend wird über die in Adressregister a11 gespeicherte Rücksprungadresse über einen indirekten Sprung in die Steuergeräte-Funktion gemäß 522 zurückgesprungen.

Die Modifikation der Programmstellen nach Methode 3 ist in Figur 6 näher dargestellt. Dabei gelten wie auch für alle übrigen Methodenbeispiele die gleichen Bezeichnungen und Abkürzungen heir speziell auch wie für Methode 2. Dabei ist mit 601 der ursprüngliche Programmcode und mit 611 der modifizierte Programmcode bezeichnet. 602 und 606 sowie 617 und 607 bilden die Steuergeräte-Funktion, hier function_a(). Dabei sind in 602 bzw. in 617 die Befehle bzw. Befehlssequenzen dargestellt und in 606 bzw. 607 die eigentliche Funktionalität. Dabei wird ein spezieller st.x (Store-Befehl) nämlich st.t betrachtet. Der Speicherbefehl st.t ist nun als 32 Bit Befehl (siehe 605) dargestellt und damit in diesem Beispiel durch einen 32Bit Funktionsaufruf call (call balcony_M3) ersetzbar. Der neue erfindungsgemäße Befehlscode bzw. Programmcode 603 (call: Funktionsaufruf) wird nun auf 605 eingespielt.

Bei der Methode 3 handelt es sich um einen 32-Bit Store-Befehl st.t in Verbindung mit einer direkten Adressierung 618 (Store Befehl mit Adresse 610). Der 32-Bit Store-Befehl wird durch einen 32-Bit Funktionsaufruf (call balcony_M3, 603) einer Software-Balkonfunktion (balcony_M3, 621) ersetzt (siehe 604). Die Software-Balkonfunktion 621 beinhaltet die Abfrage des Freischnitts und den Store-Befehl im Originalzustand. Bei aktiviertem Freischnitt wird kein Store-Befehl ausgeführt. Die Variable ist somit von der Steuergeräte-Funktion entkoppelt. Dazu erfolgt aus 612 der Aufruf 620 der Balkonfunktion 621. Über die Adresse der Steuergeräte-Variable (adr. of ecu Variable) 619 erfolgt dann der Rücksprung 622 zur Steuergeräte-Funktion.

Die Modifikation der Programmstellen nach Methode 4 ist in Figur 7 näher dargestellt. Dabei gelten wie auch für alle übrigen Methodenbeispiele die gleichen Bezeichnungen und Abkürzungen, hier speziell wie für Methode 2. Dabei ist mit 701 der ursprüngliche Programmcode und mit 711 der modifizierte Programmcode bezeichnet. 702 und 706 sowie 717 und 707 bilden die Steuergeräte-Funktion, hier function_a(). Dabei sind in 702 bzw. in 717 die Befehle bzw. Befehlssequenzen dargestellt und in 706 bzw. 707 die eigentliche Funktionalität. Mit %axx ist ein Zugriff auf ein Adressregister dargestellt und mit %dxx bz. %dyy ein Zugriff

auf ein Datenregister. Dazu sind nun die Befehle mov, st.x (Store-Befehl) call und jla wie zuvor in den Methoden beschrieben betrachtet. Der Speicherbefehl st.x ist als 32 Bit Befehl (siehe 705) dargestellt und damit in diesem Beispiel durch einen 32Bit Sprungbefehl jla ersetzbar. Der neue erfindungsgemäße Befehlscode bzw. Programmcode 703 (jla: Sprungbefehl) wird nun auf 705 eingespielt.

Bei der Methode 4 handelt es sich um einen 32-Bit Store-Befehl st.x (710) in Verbindung mit einer direkten Adressierung (718). Der 32-Bit Store-Befehl wird durch einen 32-Bit Sprung-Befehl jla (Jla balcony_M4_a) ersetzt. Der Sprung-Befehl zeigt auf die Software-Balkonfunktion1 (balcony_M4_a(), 721), die mit 720 aufgerufen wird. In der Software-Balkonfunktion1 721 wird der Inhalt des zuvor geladene Datenregisters %dxx in eine temporäre DHooks-Variable (iB_TmpMsg_xxx) zwischengespeichert. Aus 721 wird durch Funktionsaufruf call eine weitere Balkonfunktion (balcony_M4_b(), 724) mit 723 aufgerufen. Diese zweite Software-Balkonfunktion 2 beinhaltet den eigentlichen Freischnitt wie bei Methode 3. Die Software-Balkonfunktion 724 beinhaltet die Abfrage des Freischnitts. Bei deaktiviertem Freischnitt wird der Inhalt der temporären Variable iB_TmpMsg_xxx in die Steuergeräte-Variable zurückgeschrieben (siehe 725). Bei aktivem Freischnitt erfolgt kein Zurückschreiben. Die Steuergeräte-Variable ist somit von der Steuergeräte-Funktion entkoppelt. Über /22 erfolgt dann der Rücksprung zur Steuergeräte Funktion.

Die Modifikation der Funktionsaufrufe (Figur 8)

Für die lokalisierten Load-/Store-Fundstellen wird die Steuergeräte-Funktion ermittelt, in der sich die Fundstellen befinden. Dies geschieht mit dem für das Verfahren entwickelte Windows-Softwareprogramm, welches ausgehend von der Position der Load-/Store-Befehle und unter Zuhilfenahme von Referenzinformationen, die entsprechenden Anfangs- und Endadressen der Steuergeräte-Funktion ermittelt.

Anschliessend werden im gesamten Programmcode alle Funktionsaufrufe der Steuergeräte-Funktion, durch Funktionsaufrufe der neu erstellten Hook-Funktion ersetzt.

Der ursprüngliche Funktionsaufruf der Steuergeräte-Funktion erfolgt innerhalb der entsprechenden Hook-Funktion.

In Figur 8 ist aus Gründen der Übersichtlichkeit die gleiche Darstellung wie in den bisherigen Figuren 2, 4,5,6,7 gewählt um diese Modifikationsdarstellung vergleichbar zu machen. Mit 801 ist der ursprüngliche Steuergeräte-Programmcode und mit 811 der jeweils modifizierte Programmcode bezeichnet. Dabei wird eine Aufgabenliste task_list verwendet und eine entsprechende Zusatzfunktion oder Unterfunktion subfunction_x(). Der Einfachheit halber ist nun nicht mehr explizit zwischen Befehssequenz und eigentlicher Funktionalität unterschieden (vgl. dazu Figuren 3 bis 7). Mit 804 ist der Vorgang entsprechend Figur 2 zur Ermittlung der Funktionsadressen und Funktionsazfrufe bezeichnet. Gemäß 805 wird die Adresse von function_a durch die Adresse der hook_function_a ersetzt. Entsprechend wird bei 806 der Funktionsaufruf von function_a durch den Aufruf der hook_function_a ersetzt. In 807 schließlich wird der indirekte Funktionsaufruf von function_a durch einen Aufruf der hook_function_a ersetzt (hier als 32Bit Befehl). Mit nP ist dabei jeweils der durch die Ersetzungen neu gebildete Programmcode bezeichnet.

Verfahren zur Erstellung von zusätzlichem Programmcode (Figur 9)
Für jede Steuergeräte-Funktion, in der sich ein Freischnitt befindet kann somit eine Hook-Funktion angelegt werden, bzw. wird eine solche angelegt. In Figur 9 ist dazu eine schematische Darstellung solcher Hook-Funktionen hook_function_a() und hook_function_x() offenbart. Dabei ist mit 901 der Steuergeräte-Programmcode bezeichnet. 902 stellt Speicherbereich für zusätzlichen Programmcode dar. Mit 903 sind die eigentlichen Hook-Funktionen bezeichnet, in welchen mit 904 die mögliche Initialisierung von eventuell benötigten Zeigervariablen dargestellt ist. 905 offenbart den Programmcode für die Software-Freischnitte, die Konfiguration und die Anbindung insbesondere der Rapid-Prototyping-Methoden. Mit 906 schließlich ist der Aufruf der ursprünglichen Steuergeräte-Funktion function_a() dargestellt. Vergleichbar ist dies für die zweite Hook-Funktion hook_function_x() aber aus Gründen der Übersichtlichkeit nicht nochmals dargestellt.

Die Hook-Funktion beinhaltet also den Freischnitt-Mechanismus, der den Zugriff auf eine Rapid-Prototyping-Methode über Applikationsdaten steuert. Weiterhin werden in der Hook-Funktion ggf. Initialisierungen von Zeigervariablen, sowie der Funktionsaufruf der eigentlichen Steuergeräte-Funktion durchgeführt.

Nachfolgend sind Merkmale der Hook-Funktionen in Abhängigkeit der Freischnitt-Methoden dargestellt:

Zu Freischnitt-Methode 1 und 2:
Bevor ein Store-Befehl, auf eine, durch einen Zeiger adressierte Steuergeräte-Variable sinnvoll beschrieben werden kann,muss die Zeigervariable mit einer Variablen-Adresse adressiert werden. Die Initialisierung des Zeigers erfolgt in der Hook-Funktion. Ist der Freischnitt-Zugriff nicht aktiv, wird der Zeiger mit der Adresse der Steuergeräte-Variable initialisiert. Ist der Freischnitt-Zugriff aktiv, wird der Zeiger mit der Adresse einer temporären DHooks-Variable initialisiert. An dieser Stelle erfolgt z.B. bei der indirekten Adressierung der Steuergeräte-Variable, die Umlenkung des Schreibzugriffs auf die temporäre Variable.

Zu Freischnitt-Methode 3 und 4:
Bei den beiden Methoden handelt es sich um eine direkte Adressierung einer Steuergeräte-Variable. Hierbei werden keine Zeiger verwendet die initialisiert werden müssen. In der Hook-Funktion befindet sich der Mechanismus zur Steuerung des Software-Freischnittes sowie der Funktionsaufruf der ursprünglichern Steuergeräte-Funktion.

An dieser Stelle soll nochmals die Balkonfunktion bei Ersetzung durch Sprungbefehl wie bereist erläutert kurz ausgeführt werden. Der Einsatz ist vorab ausführlich dargestellt. Werden 32-Bit-breite Store-Befehle für den Software-Freischnitt herangezogen, so wird hierzu zusätzlicher Programmcode benötigt. Dieser Programmcode wird in der DHooks-Entwicklungsumgebung erzeugt und als Balkon-Funktion bezeichnet. Die Balkon-Funktionen beinhalten zusätzliche Initialisierungs-, Kopier- und Freischnitt-Mechanismen und dienen als Software-Funktionen zur Erweiterung der Freischnittfunktionalität. Balkon-Funktionen werden für die Freischnitt-Methoden 2, 3 und 4 verwendet.

Segmentierung der Speicherbereiche (Figur 10)
Für das Freischnittverfahren werden eigene Speicherbereiche des Speichermittels im Speicher-Layout S1 des Steuergeräte-Softwareprogrammes benötigt. Entsprechend Figur 10 beansprucht das Verfahren Freibereiche für Code (DHook-Code) S4, Daten (DHook-Data) S3 und RAM (DHook-RAM) S2. Im Code-Bereich liegen die Freischnitt-Funktionen (Zusatzprogrammcodes), im Datenbereich sind Applikationsgrössen abgelegt, über die der Ferischnitt-Mechanismus gesteuert wird. Die für das Verfahren benötigten Zeigervariablen und administrative RAM-Variablen sind im Freibereich für RAM-Variablen abgelegt.

Entwicklungsprozess zur Erstellung des Programmcodes (Figur 11)

Die Erzeugung des Software-Freischnittcodes erfolgt automatisch über eine für das Verfahren erstellte Entwicklungsumgebung und wird nochmals in Figur 11 verdeutlicht. Dabei kann die Variablenauswahl in Punkt 8 auch automatisch nach vorgebbaren Kriterien erfolgen bzw. durchgeführt werden.

1. Hexcode-Datei (beinhaltet den Maschinencode z.B. im Intel-Hex oder Motorola-S1S-Format)
2. Applikationsdatenbeschreibungsdatei (beinhaltet z.B. Adressen und Umrechnungsformeln von Variablen und Kenngrössen)
3. ELF-Binärdatei (Linker-Outputdatei mit Adressen von Funktionen und Variablen)
4. Programm zur Konvertierung des Maschinencodes in lesbare Assemblerbefehle
5. disassemblierter Programmcode (dient als Input für den Simulator)
6. Konverter zur Bereitstellung von Programminformationen
7. Referenzdatenbank (dient zur Auflösung offener Referenzen)
8. Anwender erstellt Auswahl von Variablen für Freischnitte
9. Informationen über freizuschneidende Steuergeräte-Variablen
10. Programmcode-Simulator (liest sequentiell alle Opcodes und überprüft die Registerinhalte
11. automatisch generierter Source-Code (beinhaltet den Programmcode für die Software-Freischnitte, Zusatzfunktionen und Informationen über die zu modifizierenden Programmcode-Stellen
12. Softwareentwicklungsumgebung (steuert alle Vorgänge zur Erzeugung des Hexcodes und der Applikationsdaten)
13. Applikationsdaten zur Steuerung der Freischnitte
14. Programmcode + Freischnittcode + Patch-Code
15. Hex/A2l-Merge-Vorgang (DynamicHooks-Anteile werden mit dem originalen Programmcode verbunden)
16. Applikationsdatenbeschreibungsdatei (beinhaltet die Projekt- sowie die Freischnitt-Applikationsdaten
17. Programmstand mit Software-Freischnitten

Verfahren zum Einbinden des Software-Freischnittcodes

Das Einbinden des Software-Freischnittcodes wird z.B. durch einen Hexcode-Merge-Lauf durchgeführt. Bei dieser Aktion werden die Ergebnisse (Hexcodes) der Entwicklungsumgebung für das Dynamische-Freischnittverfahren, in die Freibereiche des ursprünglichen Softwareprogrammes (Hexcodes) kopiert. Das Verfahren ist ähnlich aufgebaut, wie das des internen Steuergeräte-Bypass, bei dem Hexcode-Informationen aus zwei getrennten Software-Entwicklungsläufen miteinander verbunden werden.

Interner Bypass (Figur 12)

Eine Ankopplung des Bypassprogramms, das die mindestens eine Bypassfunktion umfaßt, wird durch dynamisches Linken an vorgegebenen Schnittstellen in dem Steuerprogramm durchgeführt. Auf diese Weise kann ein Zugriff auf in der elektronischen Recheneinheit vorliegende Daten während des Programmablaufs des Steuerprogramms erfolgen.

Figur 12 verdeutlicht das dynamische Linken der Bypassfunktionen. Ein erster Block 60 gibt die dem Steuerprogramm zur Verfügung stehende Vektortabelle wieder. In der Abbildung sind in der Vektortabelle 60 eine erste Spalte (Vektor) 62 und eine zweite Spalte (Kanal) 64 zu erkennen. In der ersten Spalte 62 sind Referenzen enthalten, anhand derer entschieden wird, welche der Bypassfunktionen aktiviert werden sollen. In der zweiten Spalte 64 finden sich Informationen dazu, ob es sich um einen internen oder einen externen Bypass handelt.

Ein zweiter Block 66 gibt eine Tabelle der Funktionspointer wieder, die eine Anzahl von zu aktivierenden Bypassfunktionen enthält.

Ein dritter Block 68 repräsentiert Bypassdienste. Ein vierter und ein fünfter Block 70 und 72 stehen für aktivierte Bypassfunktionen und ein sechster Block 74 für das Steuerprogramm.

In dem Steuerprogramm 74 sind potentielle Eingriffsstellen für die Bypassfunktionen vorgesehen. Dies wird durch Aktivieren einer Treiberschicht (Bypassdienste) geleistet.

Vor dem Überprüfen des Steuerprogramms 74 lädt das Applikationssystem die Bypassfunktionen in den Speicher der ECU. Außerdem werden von dem Applikationssystem Referenzen auf die Bypassfunktionen in die Tabelle 66 der Funktionspointer eingetragen.

Während der Überprüfung, d.h. während des Ablaufs des Steuerprogramms 74, entscheiden die Bypassdienste 68 in der ECU anhand von Einträgen in der Vektortabelle 60, welcher Eingriff aktiv ist, d.h. welche Bypassfunktionen aktiviert werden müssen und wo die Referenz auf die Bypassfunktion abgelegt ist, wie dies mit Pfeilen 76 verdeutlicht ist.

Ist eine Bypassfunktion für eine potentielle Eingriffstelle aktiviert, erfolgt deren Aufruf über die Referenz (dynamischer Link), was Pfeile 78 wiedergeben. Die Bypassdienste 68 führen die beiden aktiven Bypassfunktionen 70 und 72 aus (Pfeil 80). Mit den aktivierten Bypassfunktionen 70 und 72 wird auf bestimmte ausgewählte Daten 82 in dem Steuerprogramm 74 zugegriffen, was Pfeile 84 veranschaulichen.

Bei der Datenübertragung zu den Bypassfunktionen 70 und 72 greifen dieselben auf ihre Eingangsgrößen und Parameter (Applikationswerte) über Adreßreferenzen zu. Damit haben die Bypassfunktionen 70 und 72 lesenden Zugriff auf alle in der ECU statisch vorliegenden Daten 82.

Die Bypasstreiber bekommen bei der Datenübertragung des Steuerprogramms bzw. der Anwendersoftware 74 eine Referenz auf den stimulierbaren Datenfluß übergeben. Auf diese Weise kann den Anforderungen an die Datenkonsistenz entsprochen werden.

Die Berechnung des Bypasses sowie der Eingriff in den Datenfluß geschieht zu dem Zeitpunkt, an dem der Wert auch in der Basissoftware berechnet wird. Durch Anwendung dieser Triggermechanismen und der vorstehend beschriebenen Kommunikation entsteht keine Totzeit im Signalfluß der Bypassfunktion.

Softwaretest durch Stimulation der Funktionen

Bei der Erfindung handelt es sich um ein Softwaretestverfahren, mit welchem unter Anwendung der Verfahren DynamicHooks (automatischer Softwarefreischnitt) und Interner-Bypass (Rapid-Prototyping-Methode), Testabläufe nach Vorgaben des Anwenders in einer Testumgebung

generiert und innerhalb der Steuergerätesoftware zum Ablauf gebracht werden. Unter Anwendung des Software-Freischnittverfahrens DynamicHooks werden die Eingangsgrössen der entsprechenden Steuergerätefunktionen freigeschnitten. Die Testbeschreibung bzw. der zeitliche Testablauf wird in einer für das Verfahren entwickelten Programm-Beschreibungssprache innerhalb der, für das Verfahren erstellten Entwicklungsumgebung generiert. Der dabei erzeugte Sourcecode, der den Testablauf beschreibt, wird über einen Compilier- und Linkvorgang in ein lauffähiges Softwareprogramm für die Zielhardware (Steuergerät) übersetzt. Die Integration und die Ausführung des Testablaufcodes erfolgt nach dem Verfahren des Internen- Bypass.

Stimulation von Softwarefunktionen innerhalb eines Laufzeitsystems (Figur 13)

Bei der Untersuchung einer Steuergerätefunktion werden die entsprechenden Eingangsgrössen (RAM-Variablen) der Softwarefunktion freigeschnitten. Die Stimulation der freigeschnittenen RAM Grössen erfolgt über das integrierte Testablaufprogramm in der Steuergerätesoftware. Das Scheduling für den Aufruf des Testablaufcodes kann in der Entwicklungsumgebung spezifiziert werden.Der Zugriff auf die freigeschnittenen RAM-Variablen erfolgt über Referenzen.

In Figur 13 ist schematisch eine solche Stimulation dargestellt. Dabei zeigt 91 eine entsprechende Steuergeräte-Funktion bzw. Steuergeräte-Funktionen. Mit 92 (inp1, inp2, inp3) sind Schnittstellenvariablen bzw. Eingangsgrößen dargestellt, welche z.B. in anderen Software-Funktionen bzw. Funktionen berechnet werden und als globale Variable im System zur Verfügung stehen. Mit 93 sind drei der bereits erläuterten Softwarefreischnitte dargestellt, welche zweckmäßig über das Applikationssystem (insbesondere 101 aus Fig.1) aktiviert werden.

Aus der Testablaufsoftware bzw. den Stimulationsfunktionen 94 stehen die Stimulationswerte 95 (inp1s, inp2s, inp3s) bzw. deren Dateninformation als Ersatz im Rahmen des Freischnittes für die Schnittstellenvariablen bzw. Eingangsgrößen 92 zur Verfügung und werden bei Aktivierung der Freischnitte eingeblendet.

Mit 96 schließlich ist die Steuergerätesoftware selbst dargestellt.

Verfahren zur Erstellung von Testablaufsoftware (Figur 14)

In einer für das Verfahren entwickelten Testbeschreibungsprache wird der Testablauf spezifiziert. In diesem Testablauf werden die Eingangsgrössen der Steuergerätefunktion in definierte Zustände gebracht. Verschiedene Kombinationen von Stimuliwerte, Rampen oder Kurvenverläufen können spezifiziert werden. Das Zeitverhalten wird über entsprechende Softwarefunktionen innerhalb der Beschreibungssprache gesteuert.
Ein Beispiel einer solchen Testablaufbeschreibung ist in Figur 14 dargestellt.

Figur 15 zeigt zum Gegenstand der Erfindung den Entwicklungsprozess:

Mit E1 ist wiederunm die Steuergeräte-Software bezeichnet. In E2ist die Testfunktion und Informationen über die Schnittstellenvariablen. So wird hierin auch der Tesumfang also die Schnittstelleninformation der zu testenden bzw. zu stimulierenden Software-Funktion bzw. Funktion dargestellt. E3 zeigt die Freischnittauswahl, also die Auswahl der Freischnittvariablen und damit den Software-Umfang des Freischnitts. In E4 ist das Dynamik Hooks-Verfahren wie vorab beschrieben lokalisiert. In E5 ist der Programmstand mit frei geschnittenen Eingangsgrößen der zu testenden bzw. zu stimulierenden Software-Funktion, also das Steuergeräteprogramm mit den Softwarefreischnitten untergebracht.

In E6 erfolgt dann die Erstellung bzw. Abarbeitung des Testablaufskriptes, also der Testbeschreibung zum Testablauf mit einer Skriptsprache. In E7 wird durch einen Konverter bzw. Codegenerator aus den Testskript(en) übersetzbarer Code, insbesondere C-Code erstellt. Dieser erzeugte Code, also der Testablauf-Programmcode ist mit E8 dargestellt. In E9 sind Bibliotheksfunktionen in einer Bibliothek (LIB) für die Ablaufsteuerung gezeigt.

E10 zeigt nachfolgend die Software-Test-Entwicklungsumgebung zur Programmcodeerstellung also die Hexcodes und Applikationsdaten. In E10 ist ebenso das Compiler-/Linker Toolset untergebracht, wodurch schließlich die Software für den Testablauf im Steuergerät in E10 erzeugt wird. Dabei wird zwischen E11, den Applikationsdaten (A2L) zur Steuerung des Testablaufs und der Variablenzugriffe sowie E12 dem Programmcode und Testablaufcode, also der eigentichen Testablaufsoftware (HEX) unterschieden.

Das Zusammenführen, der Merge-Vorgang (A2l-HEX-Merge-Vorgang) erfolgt in E13 zur Erzeugung der Projekt-Software mit integrierten Softwarefreischnitten und dem Testablaufcode.

Daraus entsteht schließlich das Steuergeräteprogramm mit dem integrierten Softwaretestablauf. Darin enthalten ist mit E14 die Applikationsdatenbeschreibungsdatei mit den Projekt- bzw. Freischnittdaten und den Applikationsdaten für die Testablaufsteuerung. Ebenfalls enthalten ist mit E15 der fertige Programmstand mit den Softwarefreischnitten und dem Testablaufcode.

Damit ist es erfindungsgemäß möglich durch ein integriertes Softwaretestverfahren und die entsprechende Vorrichtung Softwarefunktionen bzw. Funktionen zur Steuerung von Betriebsabläufen, insbesondere bei einem Fahrzeug, innerhalb eines Steuergerätesystems zur Laufzeit durch Stimulation deren Eingangsgrößen über entsprechende Softwareeingriffe bzw. Softwarefreischnitte unter verschiedenen Betriebspunkten zu testen.

Ansprüche

1. Verfahren zur Stimulation von Funktionen zur Steuerung von Betriebsabläufen, wobei die Funktionen auf wenigstens eine globale Variable wenigstens eines Programms zur Steuerung zurückgreift, dadurch gekennzeichnet, dass wenigstens eine Stimulationsfunktion vorgesehen ist, die auf die wenigstens eine globale Variable über wenigstens einen Softwarefreischnitt zugreift.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die Stimulation der Funktionen innerhalb des Programms zur Laufzeit des Programms erfolgt.

3. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die Stimulation der Funktionen in Echtzeit innerhalb eines Laufzeitsystems erfolgt.

4. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass der globalen Variable eine erste Dateninformation zugeordnet ist und diese erste Dateninformation durch eine zweite Dateninformation, die neuen Stimulationswerten entspricht ersetzt wird.

5. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die Stimulation der Funktionen durch einen internen Bypass erfolgt.

6. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass der Softwarefreischnitt dadurch erfolgt, dass der globalen Variable eine Adressinformation zugeordnet ist, wobei die Adressinformation durch einen Ladebefehl aus einem Speichermittel geladen wird und die

Adressinformation der globalen Variable des Ladebefehls ersetzt wird.

7. Verfahren nach Anspruch 6, dadurch gekennzeichnet, dass die Adressinformation der globalen Variable durch die Adressinformation einer Zeigervariable ersetzt wird.

8. Verfahren nach Anspruch 6, dadurch gekennzeichnet, dass aus der Adressinformation eine Anfangsadresse der Funktion ermittelt wird.

9. Verfahren nach Anspruch 6, dadurch gekennzeichnet, dass die Funktionen zur Steuerung von Betriebsabläufen durch Ersetzen der Adressinformation durch Zusatzfunktionen ersetzt werden.

10. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass der Softwarefreischnitt dadurch erfolgt, dass die globale Variable durch einen Speicherbefehl angesprochen wird und der Speicherbefehl auf die globale Variable manipuliert wird, indem der Speicherbefehl durch einen Sprungbefehl ersetzt wird.

11. Verfahren nach Anspruch 10, dadurch gekennzeichnet, dass die Funktionen zur Steuerung der Betriebsabläufe durch Ersetzen des Speicherbefehls durch den Sprungbefehl durch Zusatzfunktionen ersetzt werden.

12. Vorrichtung zur Stimulation von Funktionen zur Steuerung von Betriebsabläufen, wobei die Funktionen auf wenigstens eine globale Variable wenigstens eines Programms zur Steuerung zurückgreift, dadurch gekennzeichnet, dass erste Stimulationsmittel vorgesehen sind, die derart ausgestaltet sind, dass wenigstens eine Stimulationsfunktion aktiviert wird und Kontrollmittel vorgesehen sind, die derart ausgestaltet sind dass diese wenigstens einen Softwarefreischnitt erzeugen, wobei die Stimulationsfunktion auf die wenigstens eine globale Variable über den Softwarefreischnitt zugreift.

13. Steuergerät mit einer Vorrichtung zur Stimulation von Funktionen zur Steuerung von Betriebsabläufen, wobei die Funktionen auf wenigstens eine globale Variable wenigstens eines Programms zur Steuerung zurückgreift, dadurch gekennzeichnet, dass erste Stimulationsmittel vorgesehen sind, die derart ausgestaltet sind, dass wenigstens eine Stimulationsfunktion aktiviert wird und Kontrollmittel vorgesehen sind, die derart

ausgestaltet sind dass diese wenigstens einen Softwarefreischnitt erzeugen, wobei die Stimulationsfunktion auf die wenigstens eine globale Variable über den Softwarefreischnitt zugreift.

14.Computerprogramm-Produkt mit Programmcode, der auf einem maschinenlesbaren Träger gespeichert ist, zur Durchführung des Verfahrens nach einem der Ansprüche 1 bis 11, wenn das Programm auf einem Computer ausgeführt wird

15.Computerprogramm mit Programmcode zur Durchführung aller Schritte nach einem der Ansprüche 1 bis 11, wenn das Programm in einem Computer ausgeführt wird.

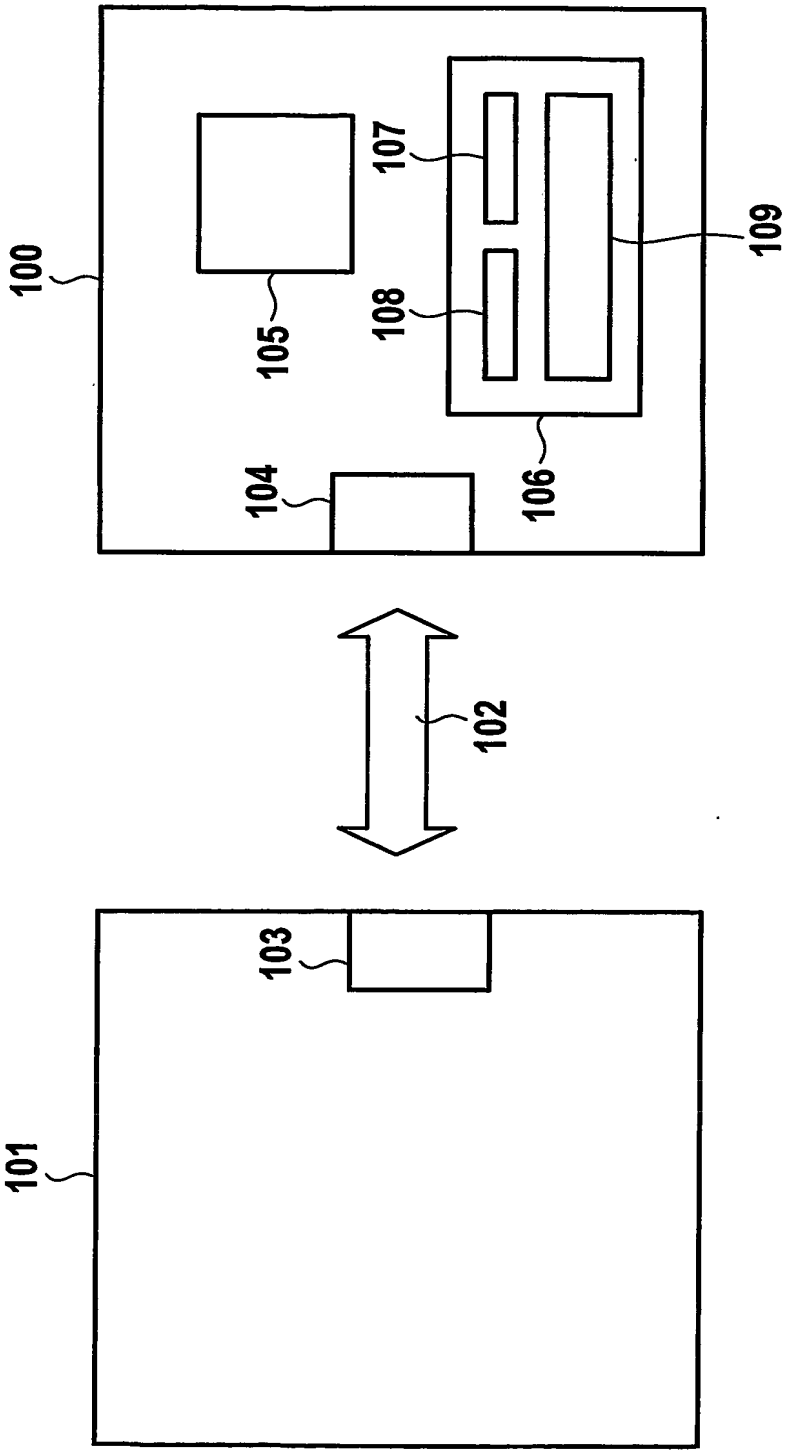


Fig. 1

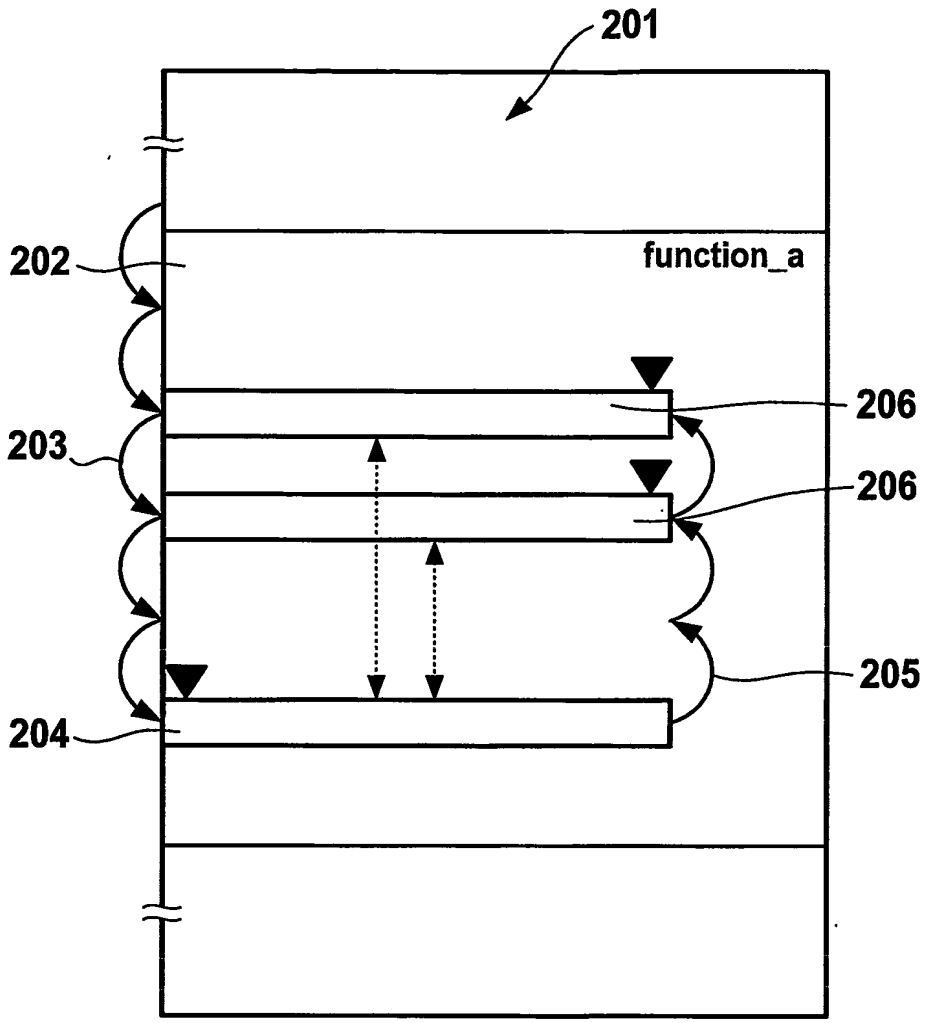
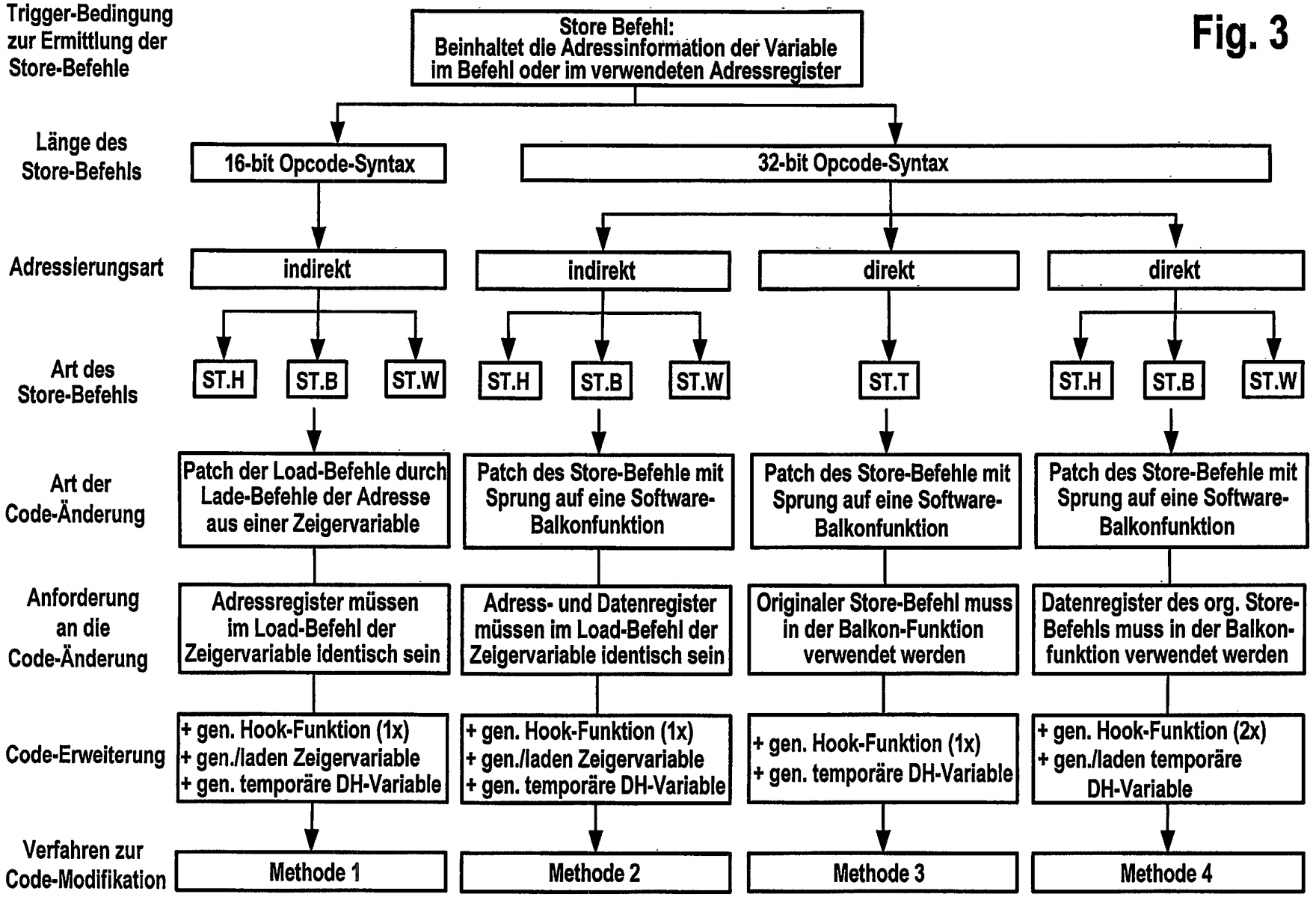


Fig. 2

Ersatzblatt

Fig. 3



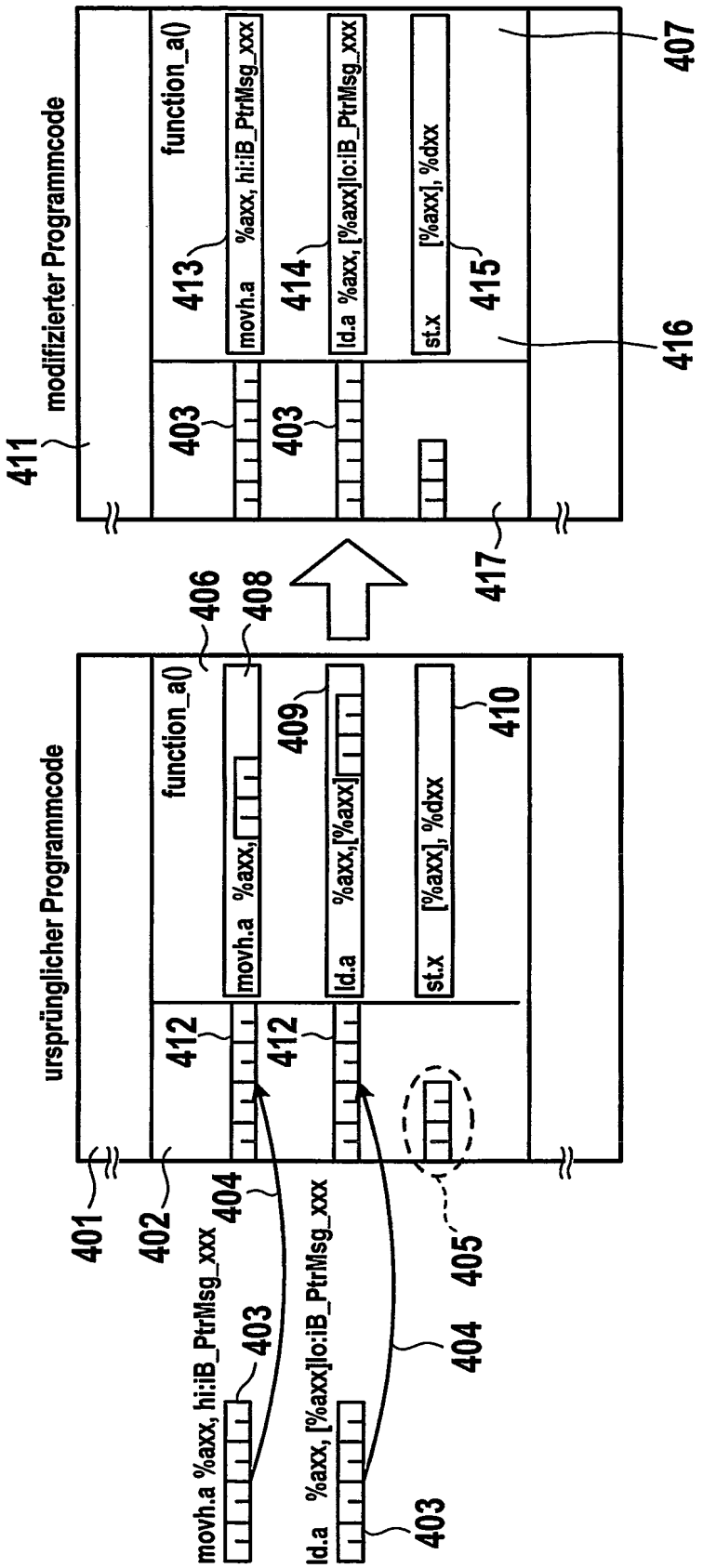

ursprünglicher Programmcode
modifizierter Programmcode
function_a()
movh.a %axx, hi:iB_PtrMsg_xxx
ld.a %axx, [%axx]lo:iB_PtrMsg_xxx
st.x [%axx], %dxx
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417

Fig. 4

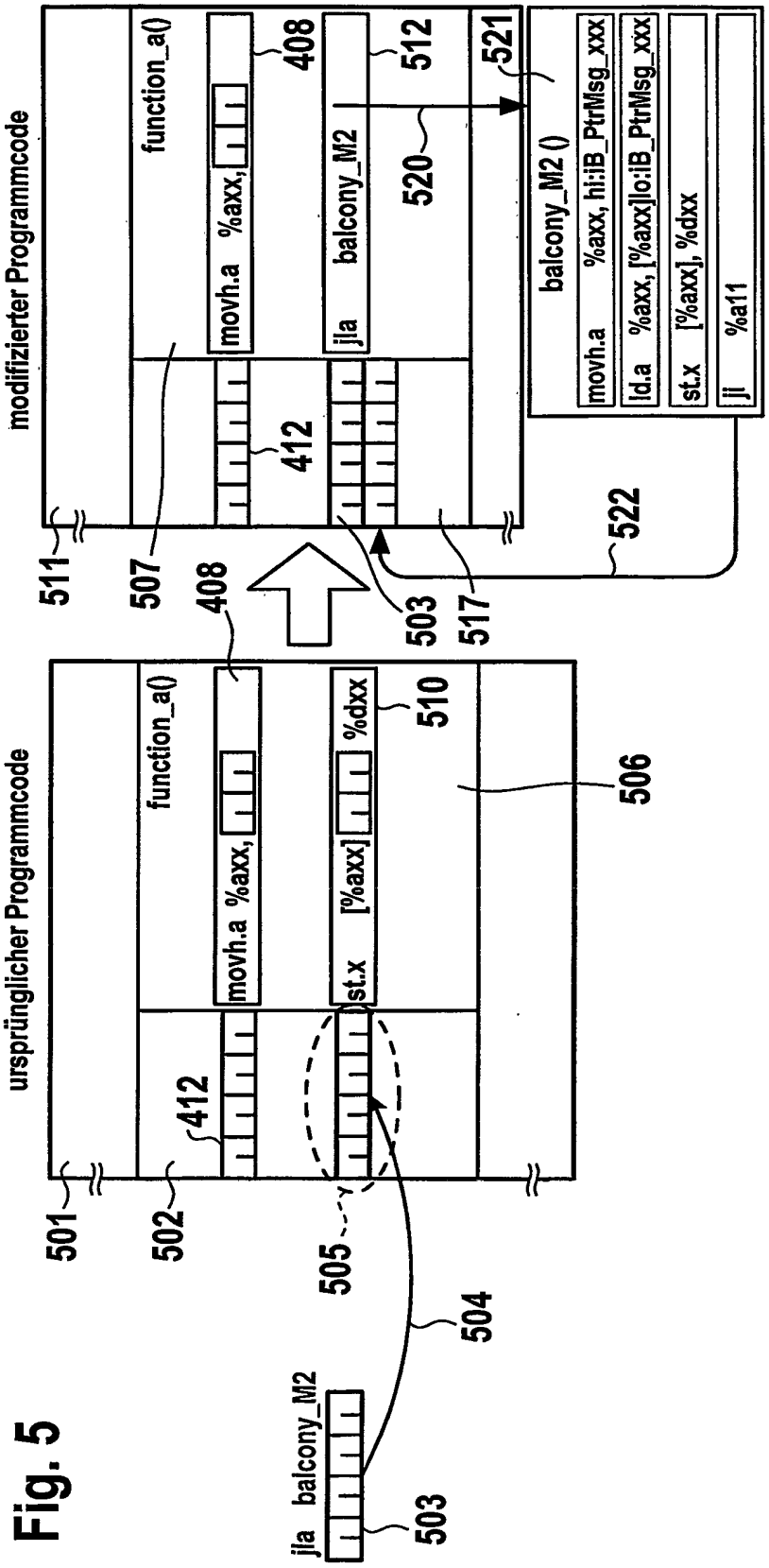

modifizierter Programmcode
function_a()
movh.a %axx,
408
jla balcony_M2
512
520
521
balcony_M2 ()
movh.a %axx, hi:iB_PtrMsg_xxx
ld.a %axx, [%axx]lo:iB_PtrMsg_xxx
st.x [%axx], %dxx
ji %a11
412
511
507
408
503
517
522
ursprünglicher Programmcode
function_a()
movh.a %axx,
st.x [%axx]
%dxx
510
506
412
501
502
505
504
jla balcony_M2
503

Fig. 5

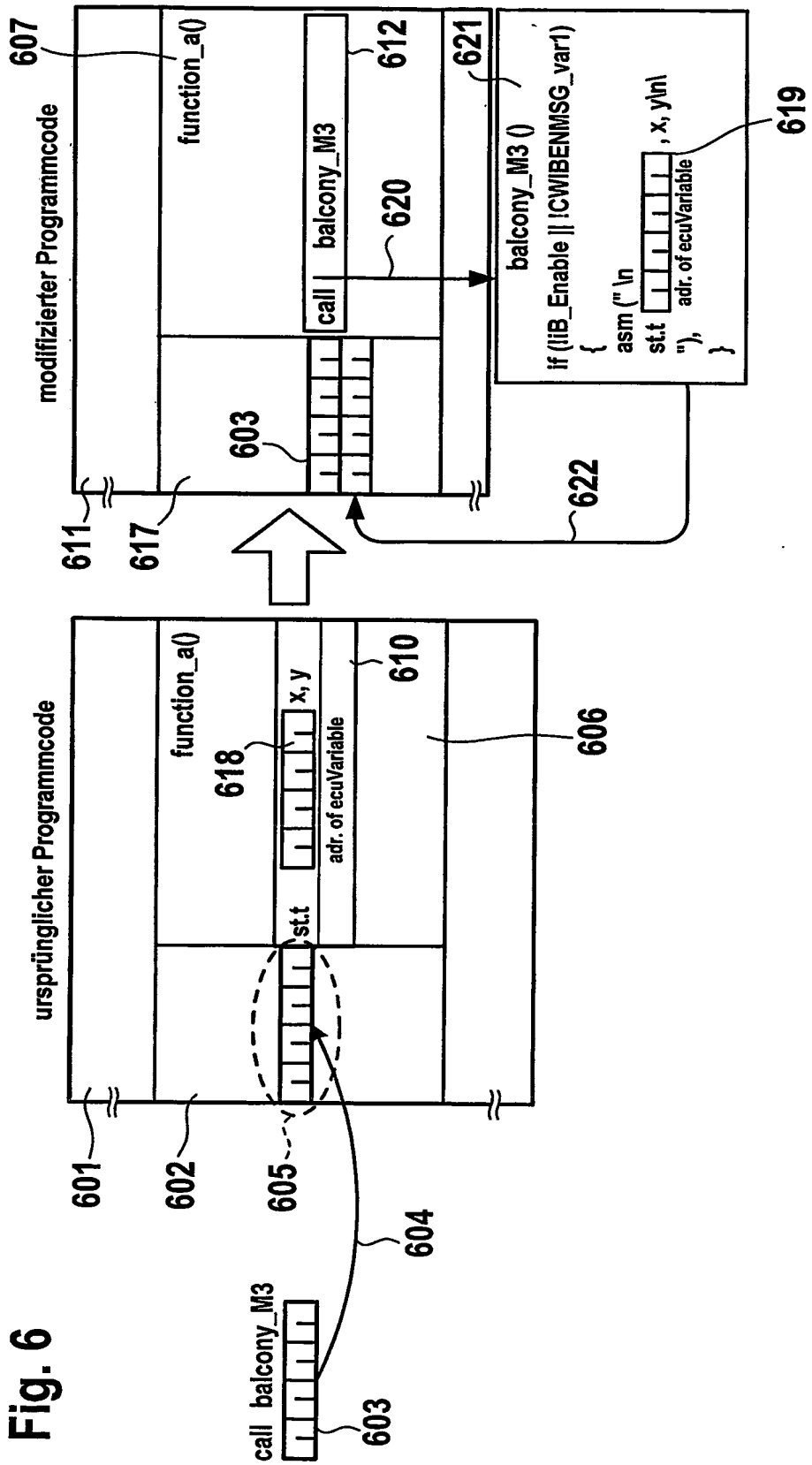

607
modifizierter Programmcode
function_a()
call balcony_M3
612
621
620
603
611
617
622
balcony_M3 ()
if (!iB_Enable || ICWIBENMSG_var1)
{
asm (" \n
st.t
"),
}
, x, y\n\
adr. of ecuVariable
619
ursprünglicher Programmcode
function_a()
618
x, y
st.t
adr. of ecuVariable
610
606
601
602
605
604
call balcony_M3
603

Fig. 6

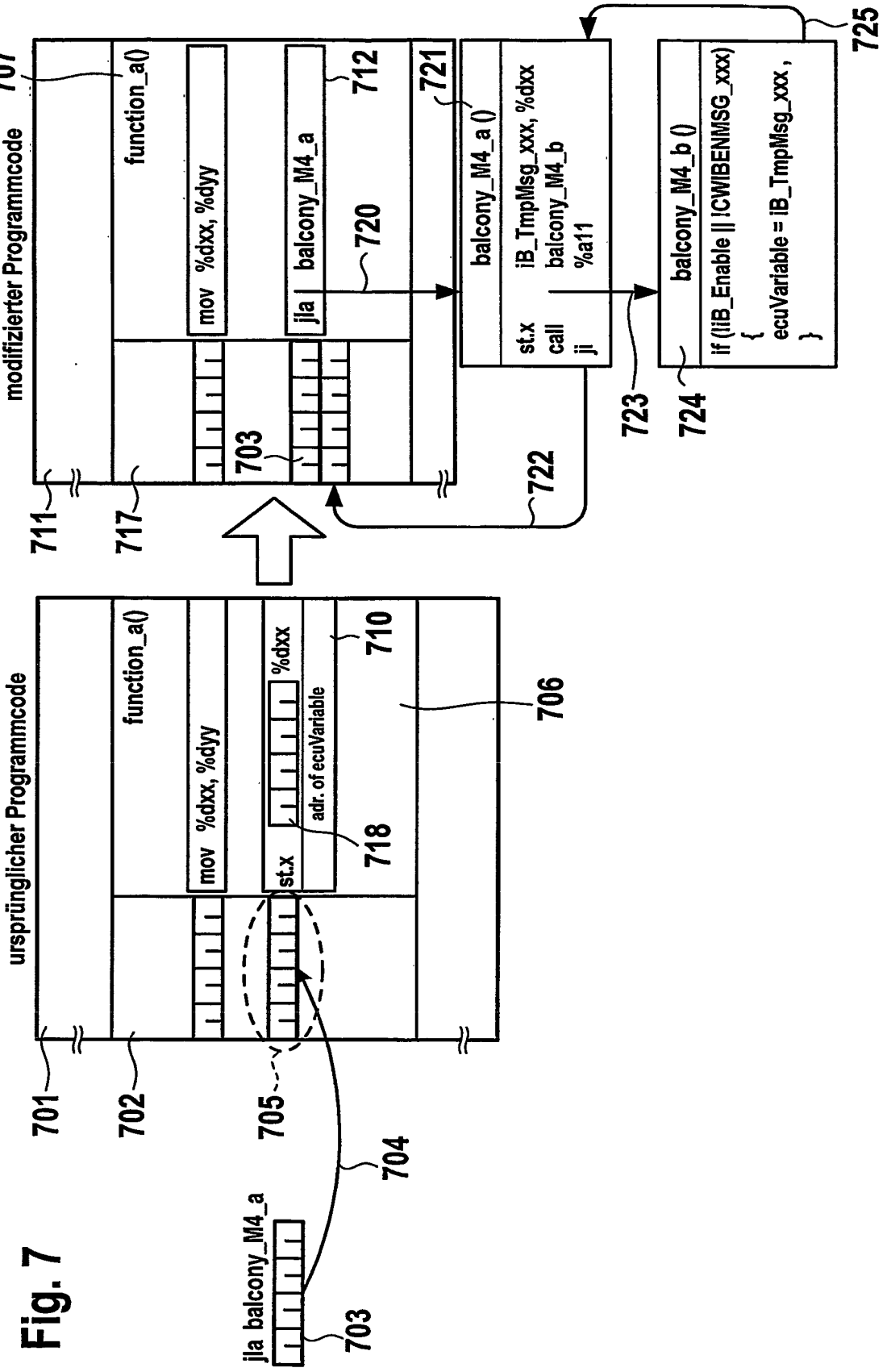

ursprünglicher Programmcode
function_a()
mov %dxx, %dyy
st.x %dxx
adr. of ecuVariable
jla balcony_M4_a
modifizierter Programmcode
function_a()
mov %dxx, %dyy
jla balcony_M4_a
balcony_M4_a ()
st.x iB_TmpMsg_xxx, %dxx
call balcony_M4_b
ji %a11
balcony_M4_b ()
if (!iB_Enable || !CWIBENMSG_xxx)
{
ecuVariable = iB_TmpMsg_xxx ,
}
701
702
703
704
705
706
707
710
711
712
717
718
720
721
722
723
724
725

Fig. 7

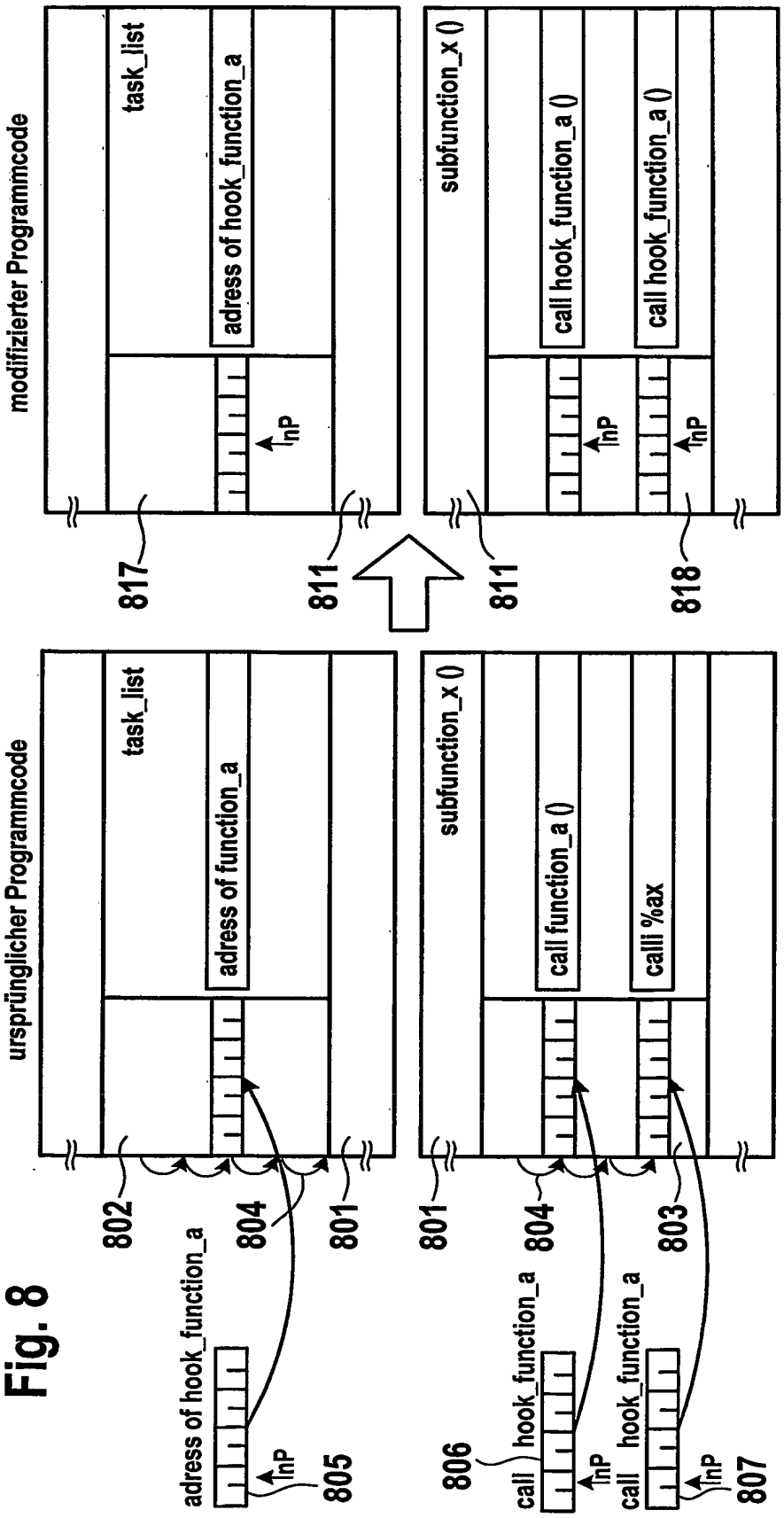

ursprünglicher Programmcode
task_list
adress of function_a
subfunction_x ()
call function_a ()
calli %ax
802
804
801
801
804
803
adress of hook_function_a
call hook_function_a
call hook_function_a
lnP
805
806
807
modifizierter Programmcode
task_list
adress of hook_function_a
subfunction_x ()
call hook_function_a ()
call hook_function_a ()
817
811
811
818

Fig. 8

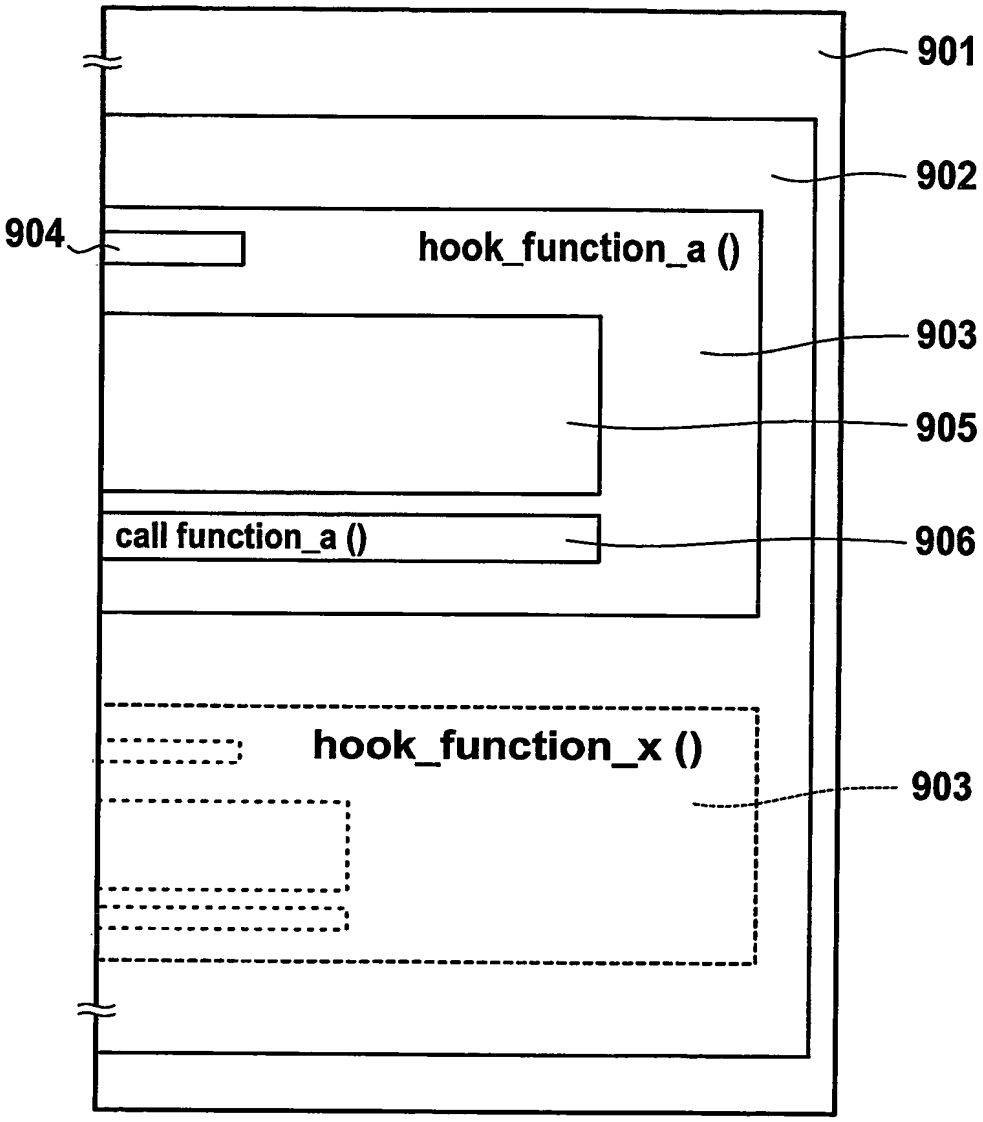

901
902
904
hook_function_a ()
903
905
call function_a ()
906
hook_function_x ()
903

Fig. 9

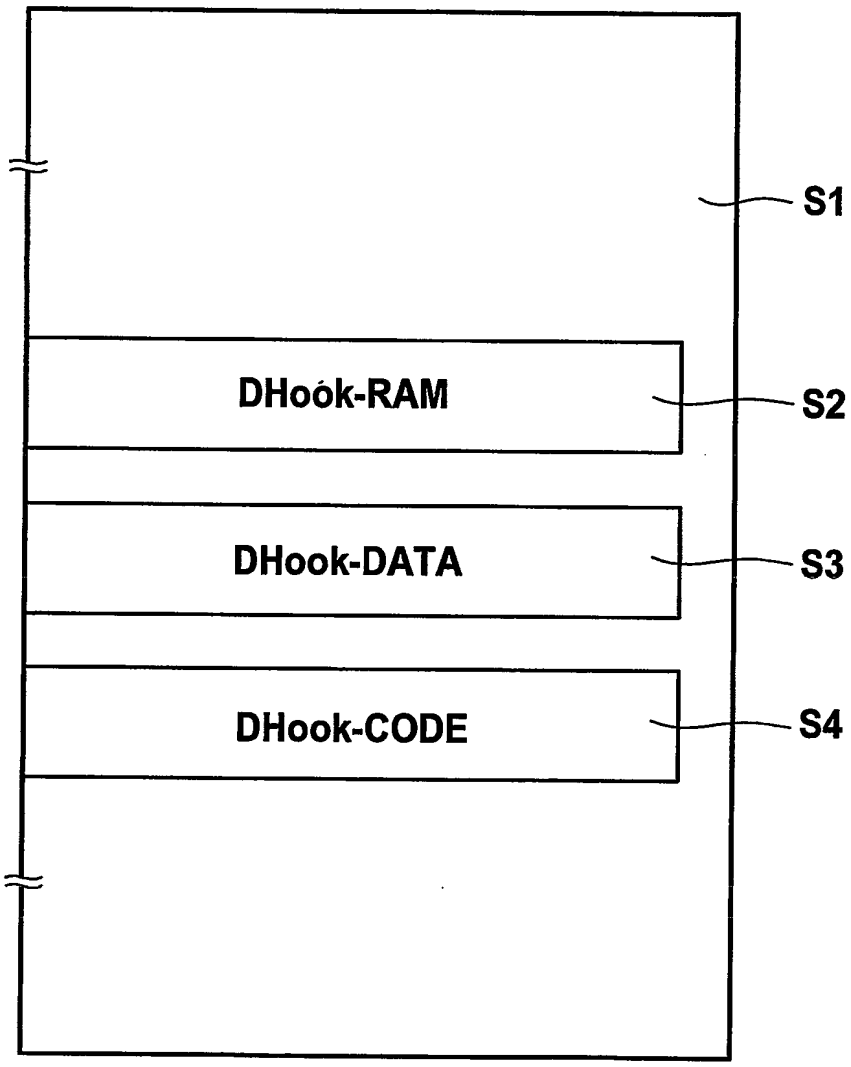


Fig. 10

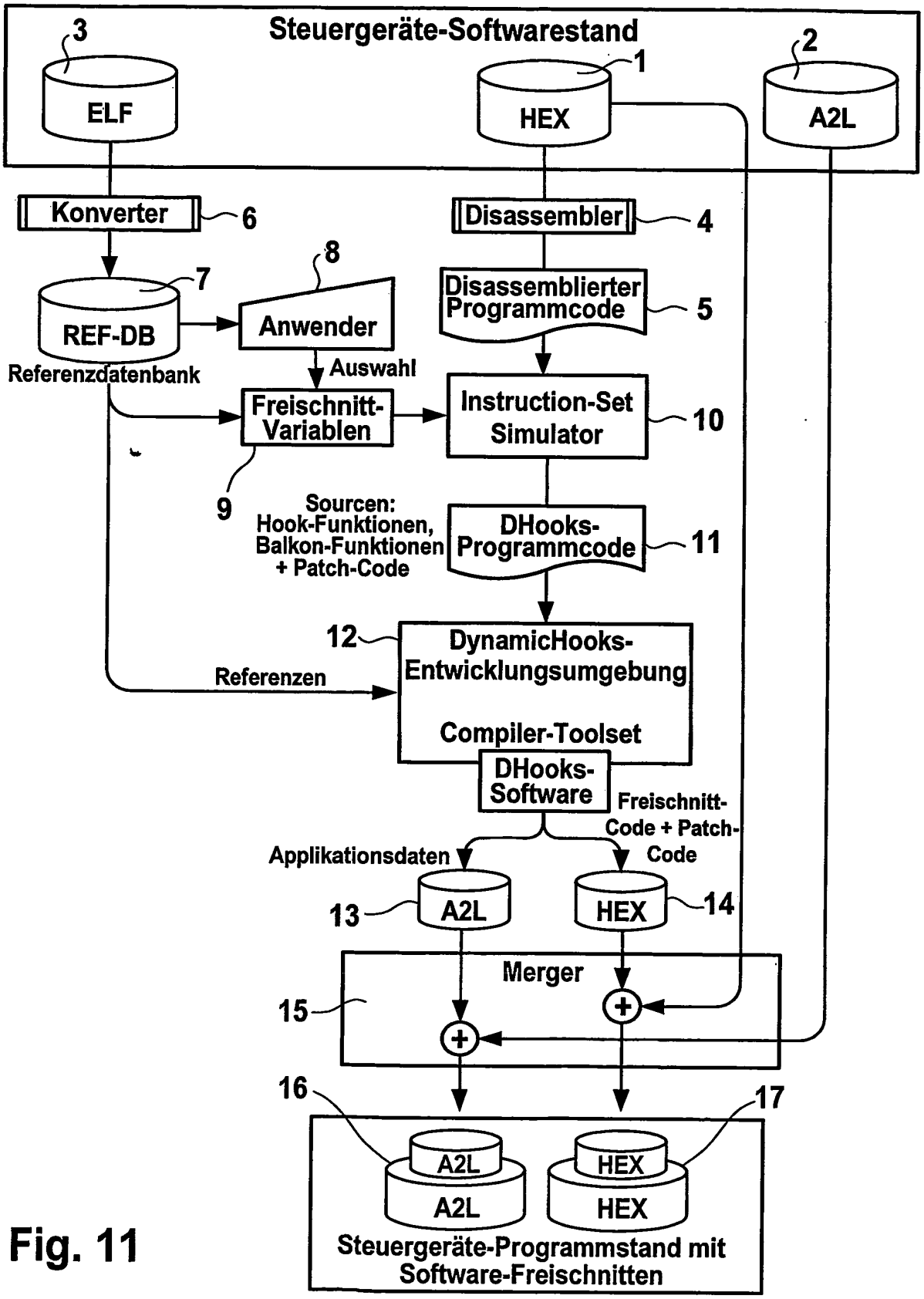
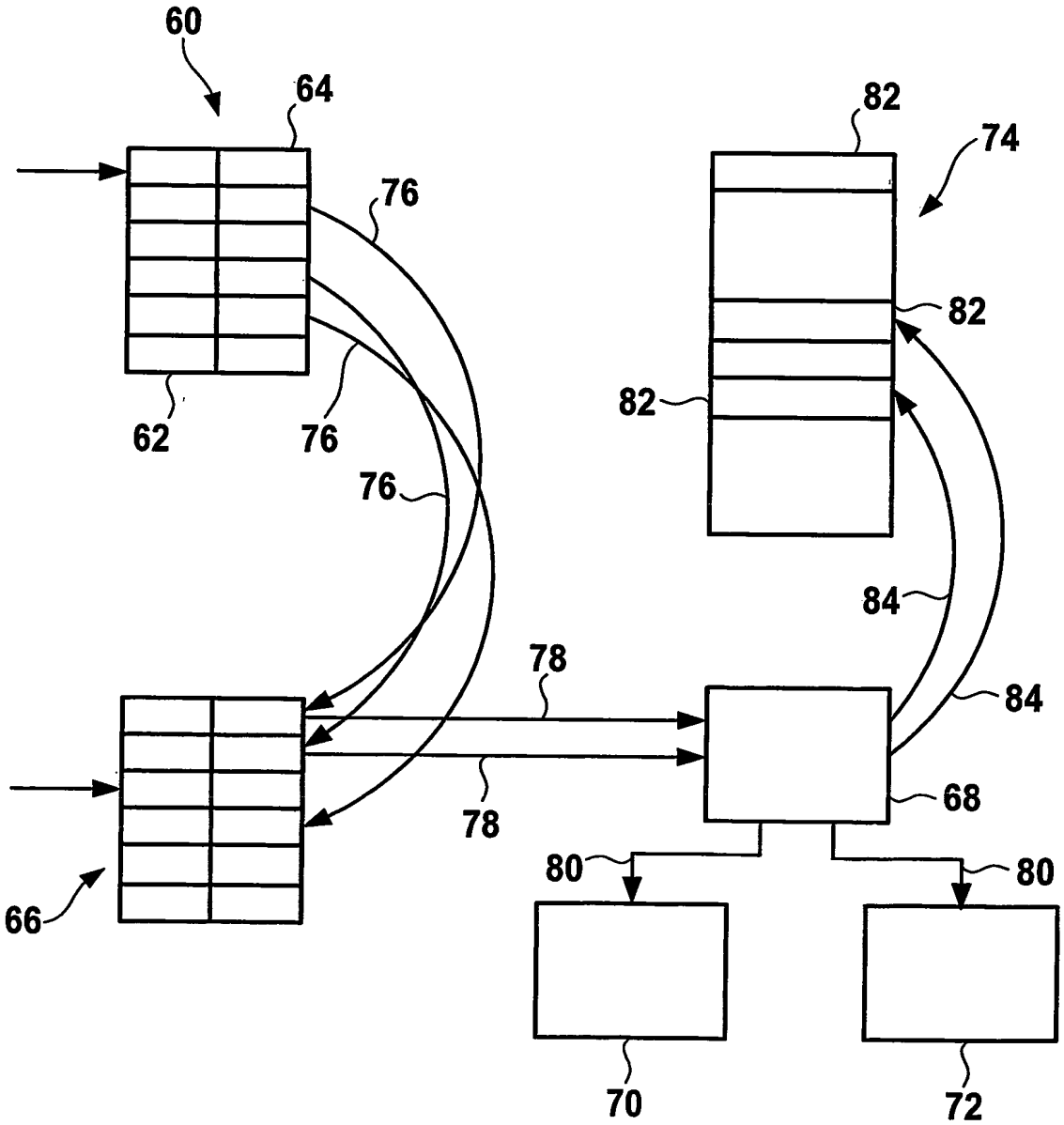

Steuergeräte-Softwarestand
3
ELF
1
HEX
2
A2L
Konverter
6
Disassembler
4
8
7
REF-DB
Anwender
Disassemblierter Programmcode
5
Referenzdatenbank
Auswahl
Freischnitt-Variablen
Instruction-Set Simulator
10
9
Sourcen: Hook-Funktionen, Balkon-Funktionen + Patch-Code
DHooks-Programmcode
11
12
DynamicHooks-Entwicklungsumgebung
Referenzen
Compiler-Toolset
DHooks-Software
Freischnitt-Code + Patch-Code
Applikationsdaten
13
A2L
HEX
14
Merger
15
16
17
A2L
A2L
HEX
HEX
Steuergeräte-Programmstand mit Software-Freischnitten

Fig. 11

Fig. 12



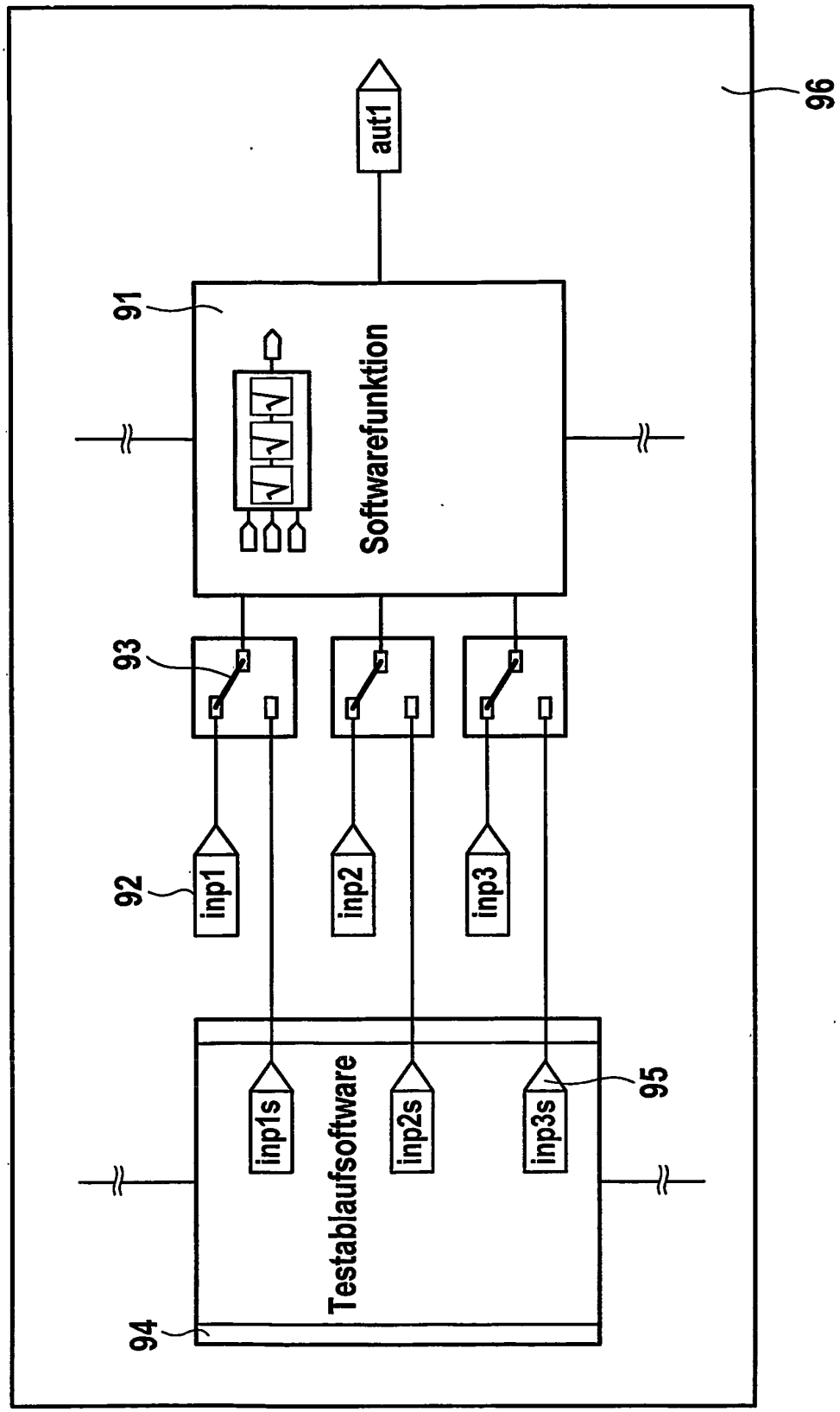

aut1
96
91
Softwarefunktion
93
92
inp1
inp2
inp3
inp1s
inp2s
inp3s
95
Testablaufsoftware
94

Fig. 13

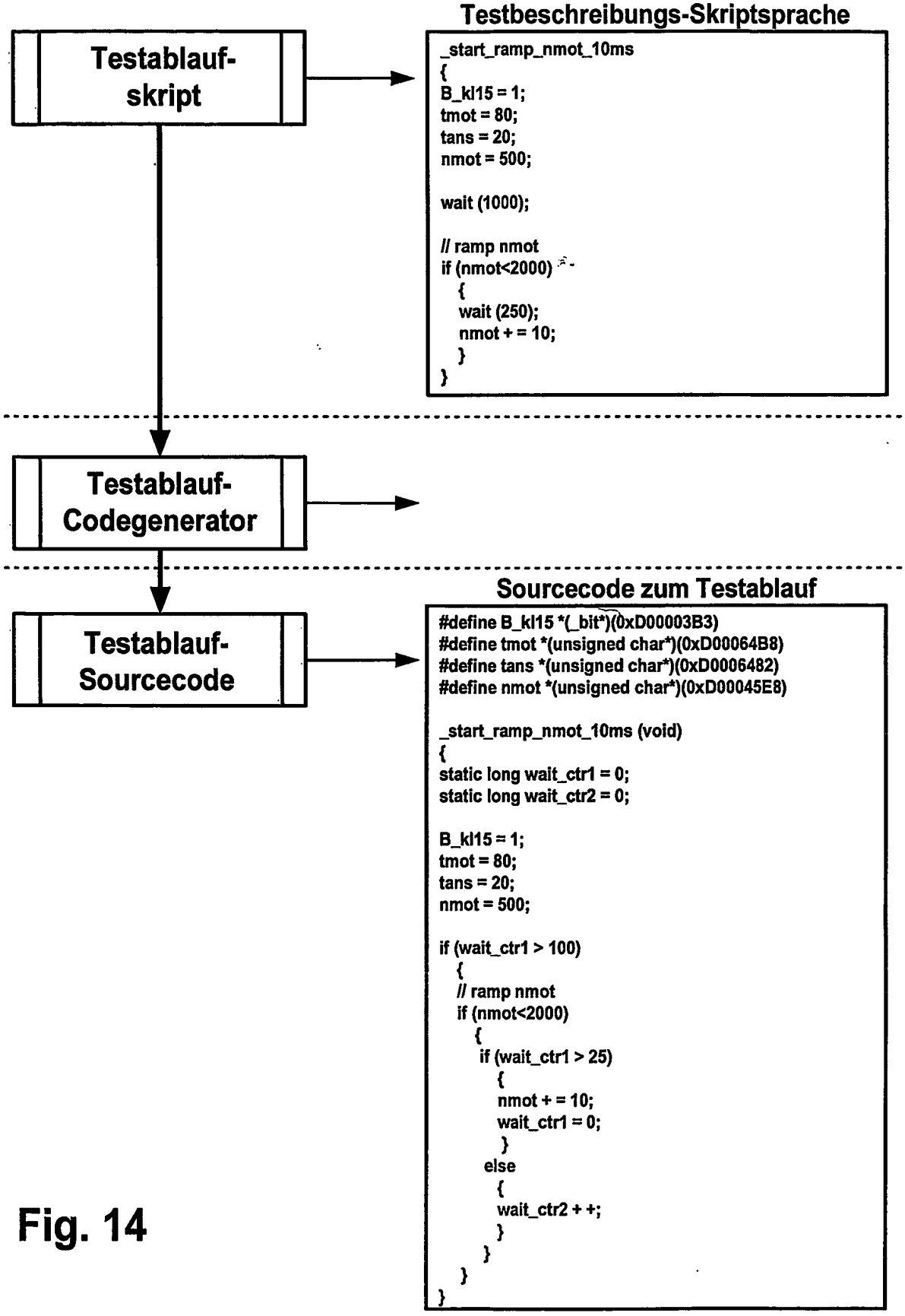
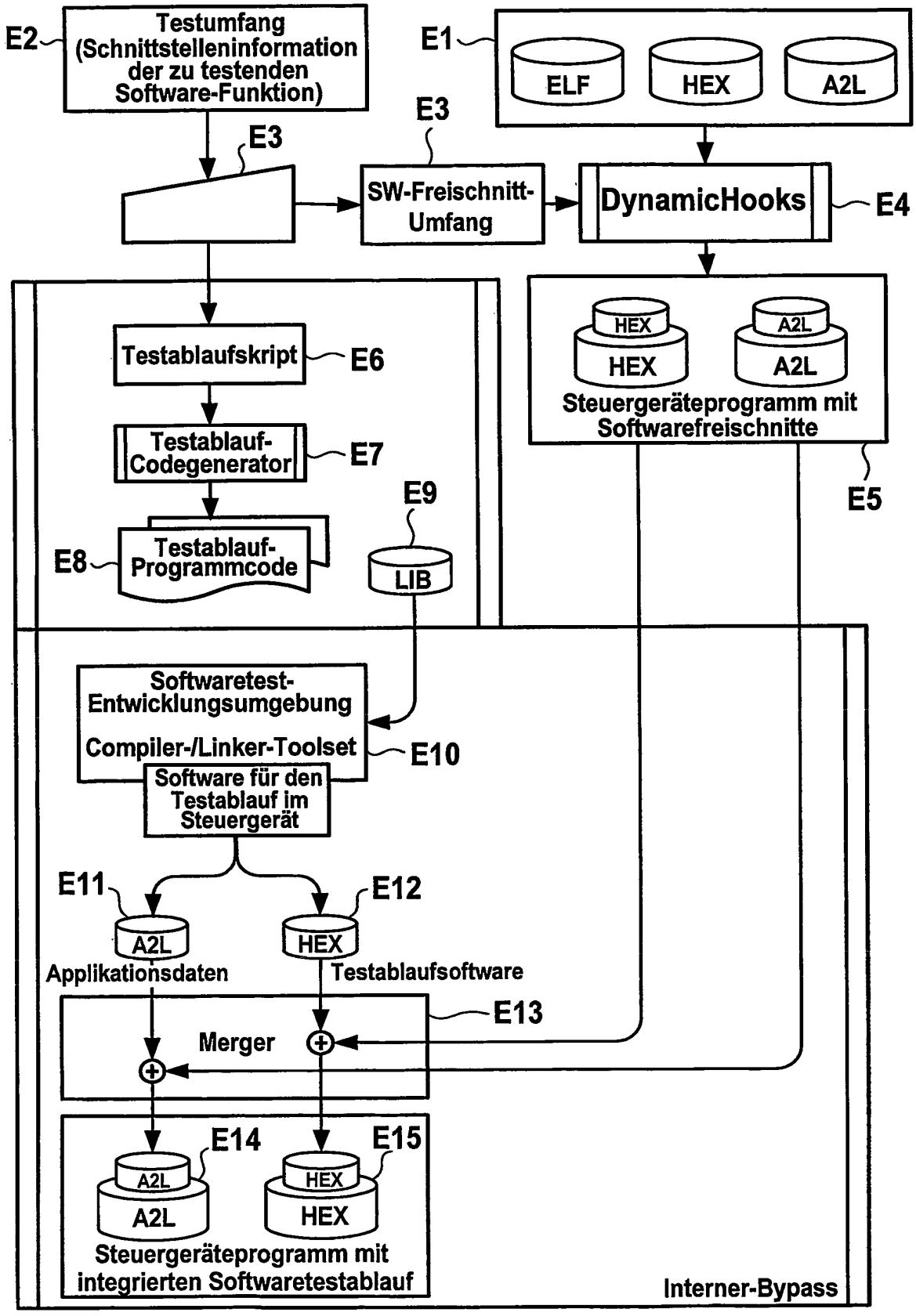

Testablauf-skript
Testbeschreibungs-Skriptsprache
_start_ramp_nmot_10ms
{
B_kl15 = 1;
tmot = 80;
tans = 20;
nmot = 500;
wait (1000);
// ramp nmot
if (nmot<2000)
{
wait (250);
nmot + = 10;
}
}
Testablauf-Codegenerator
Testablauf-Sourcecode
Sourcecode zum Testablauf
#define B_kl15 *(_bit*)(0xD00003B3)
#define tmot *(unsigned char*)(0xD00064B8)
#define tans *(unsigned char*)(0xD0006482)
#define nmot *(unsigned char*)(0xD00045E8)
_start_ramp_nmot_10ms (void)
{
static long wait_ctr1 = 0;
static long wait_ctr2 = 0;
B_kl15 = 1;
tmot = 80;
tans = 20;
nmot = 500;
if (wait_ctr1 > 100)
{
// ramp nmot
if (nmot<2000)
{
if (wait_ctr1 > 25)
{
nmot + = 10;
wait_ctr1 = 0;
}
else
{
wait_ctr2 + +;
}
}
}
}

Fig. 14

Fig. 15


E2
Testumfang (Schnittstelleninformation der zu testenden Software-Funktion)
E1
ELF
HEX
A2L
E3
E3
SW-Freischnitt-Umfang
DynamicHooks
E4
HEX
A2L
HEX
A2L
Steuergeräteprogramm mit Softwarefreischnitte
E5
Testablaufskript
E6
Testablauf-Codegenerator
E7
E9
E8
Testablauf-Programmcode
LIB
Softwaretest-Entwicklungsumgebung
Compiler-/Linker-Toolset
E10
Software für den Testablauf im Steuergerät
E11
E12
A2L
HEX
Applikationsdaten
Testablaufsoftware
E13
Merger
E14
E15
A2L
HEX
A2L
HEX
Steuergeräteprogramm mit integrierten Softwaretestablauf
Interner-Bypass

INTERNATIONAL SEARCH REPORT

International Application No
PCT/DE2004/002462

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G05B15/02 G06F11/36

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G05B G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4 866 665 A (HASWELL-SMITH ET AL) 12 September 1989 (1989-09-12) the whole document	1-15
X	INTERNATIONAL BUSINESS MACHINES CORPORATION: "Method for allowing multiple breakpoints on a processor with a single breakpoint address register" RESEARCH DISCLOSURE, KENNETH MASON PUBLICATIONS, HAMPSHIRE, GB, vol. 462, no. 142, October 2002 (2002-10), XP007131439 ISSN: 0374-4353 the whole document	1,12-15
A	EP 0 577 393 A (CANON KABUSHIKI KAISHA) 5 January 1994 (1994-01-05) the whole document	

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 March 2005	31/03/2005

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Kuntz, J-M

Form PCT/ISA/210 (second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/DE2004/002462

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HOLLINGSWORTH J K ET AL: "MDL: a language and compiler for dynamic program instrumentation" PARALLEL ARCHITECTURES AND COMPILATION TECHNIQUES., 1997. PROCEEDINGS., 1997 INTERNATIONAL CONFERENCE ON SAN FRANCISCO, CA, USA 10-14 NOV. 1997, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 10 November 1997 (1997-11-10), pages 201-212, XP010261151 ISBN: 0-8186-8090-3 the whole document	
A	JEZIERNY R D ED - INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: "Software error reporting and recovery techniques used in the EWSD digital switching system coordination processor from Siemens Stromberg-Carlson" COMMUNICATIONS - RISING TO THE HEIGHTS. DENVER, JUNE 23 - 26, 1991, PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, NEW YORK, IEEE, US, vol. VOL. 1, 23 June 1991 (1991-06-23), pages 1154-1158, XP010044197 ISBN: 0-7803-0006-8 the whole document	

INTERNATIONAL SEARCH REPORT

[redacted] Information on patent family members

International Application No
[redacted]/DE2004/002462

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4866665 A	12-09-1989	GB 2197506 A	18-05-1988
		DE 3732808 A1	26-05-1988
		FR 2606903 A1	20-05-1988
		JP 63115245 A	19-05-1988
EP 0577393 A	05-01-1994	JP 2922723 B2	26-07-1999
		JP 6019740 A	28-01-1994
		DE 69327551 D1	17-02-2000
		DE 69327551 T2	25-05-2000
		EP 0577393 A1	05-01-1994
		US 5481705 A	02-01-1996

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/DE2004/002462

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
IPK 7 G05B15/02 G06F11/36

Nach der Internationalen Patentklassifikation (IPK) oder nach der nationalen Klassifikation und der IPK

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
IPK 7 G05B G06F

Recherchierte aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	US 4 866 665 A (HASWELL-SMITH ET AL) 12. September 1989 (1989-09-12) das ganze Dokument	1-15
X	INTERNATIONAL BUSINESS MACHINES CORPORATION: "Method for allowing multiple breakpoints on a processor with a single breakpoint address register" RESEARCH DISCLOSURE, KENNETH MASON PUBLICATIONS, HAMPSHIRE, GB, Bd. 462, Nr. 142, Oktober 2002 (2002-10), XP007131439 ISSN: 0374-4353 das ganze Dokument	1,12-15
A	EP 0 577 393 A (CANON KABUSHIKI KAISHA) 5. Januar 1994 (1994-01-05) das ganze Dokument	

-/--

[X] Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen

[X] Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :
"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist
"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist
"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)
"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht
"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist
"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist
"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden
"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist
"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
18. März 2005	31/03/2005

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Kuntz, J-M

Formblatt PCT/ISA/210 (Blatt 2) (Januar 2004)

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/DE2004/002462

C.(Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie°	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
A	HOLLINGSWORTH J K ET AL: "MDL: a language and compiler for dynamic program instrumentation" PARALLEL ARCHITECTURES AND COMPILATION TECHNIQUES., 1997. PROCEEDINGS., 1997 INTERNATIONAL CONFERENCE ON SAN FRANCISCO, CA, USA 10-14 NOV. 1997, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 10. November 1997 (1997-11-10), Seiten 201-212, XP010261151 ISBN: 0-8186-8090-3 das ganze Dokument ----	
A	JEZIERNY R D ED - INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: "Software error reporting and recovery techniques used in the EWSD digital switching system coordination processor from Siemens Stromberg-Carlson" COMMUNICATIONS - RISING TO THE HEIGHTS. DENVER, JUNE 23 - 26, 1991, PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, NEW YORK, IEEE, US, Bd. VOL. 1, 23. Juni 1991 (1991-06-23), Seiten 1154-1158, XP010044197 ISBN: 0-7803-0006-8 das ganze Dokument ----	

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen
PCT/DE2004/002462

Im Recherchenbericht angeführtes Patentdokument		Datum der Veröffentlichung	Mitglied(er) der Patentfamilie		Datum der Veröffentlichung
US 4866665	A	12-09-1989	GB	2197506 A	18-05-1988
			DE	3732808 A1	26-05-1988
			FR	2606903 A1	20-05-1988
			JP	63115245 A	19-05-1988
EP 0577393	A	05-01-1994	JP	2922723 B2	26-07-1999
			JP	6019740 A	28-01-1994
			DE	69327551 D1	17-02-2000
			DE	69327551 T2	25-05-2000
			EP	0577393 A1	05-01-1994
			US	5481705 A	02-01-1996